(1)

DESIGN OF A SYNTHETIC APERTURE ARRAY TO SUPPORT EXPERIMENTS IN ACTIVE CONTROL OF SCATTERING

by

JAMES P. DULLEA

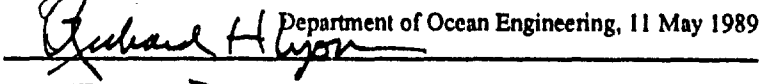
B.N.E. GEORGIA INSTITUTE OF TECHNOLOGY
(1976)
M.S.S.M UNIVERSITY OF SOUTHERN CALIFORNIA
(1983)

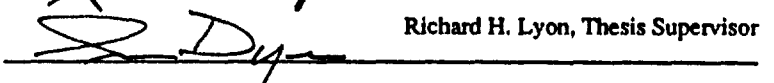
Submitted to the Department of Ocean Engineering
in Partial Fulfillment of the
Requirements for the Degree of

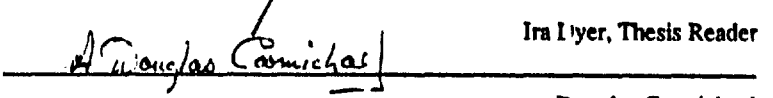
NAVAL ENGINEER
and
MASTER OF SCIENCE IN MECHANICAL ENGINEERING
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
JUNE 1990

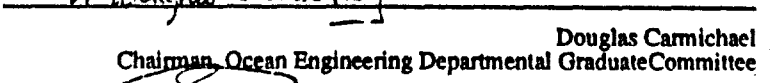
N00123-89-G-0580

Signature of Author 
Department of Ocean Engineering, 11 May 1989

Certified by 
Richard H. Lyon, Thesis Supervisor

Certified by 
Ira Dyer, Thesis Reader

Accepted by 
Douglas Carmichael
Chairman, Ocean Engineering Departmental GraduateCommittee

Accepted by 
Ain Sonin
Chairman, Mechanical Engineering Departmental Graduate Committee

DESIGN OF A SYNTHETIC APERTURE ARRAY TO SUPPORT EXPERIMENTS IN ACTIVE CONTROL OF SCATTERING

by

JAMES P. DULLEA

Submitted to the Departments of Ocean Engineering and Mechanical Engineering on 11 May 1990 in partial fulfillment of the requirements for the Degrees of Naval Engineer and Master of Science in Mechanical Engineering

Abstract

A synthetic aperture acoustic array is used to as a spatial filter in a reverberant test facility. The array is used to examine the c[illegible]nge in the received signal when a fluid loaded steel plate is replaced by the free surface. Twenty millisecond active pulses at 2 kHz were scattered by the target plate and recorded by the synthetic aperture array for post processing. The method of reverberant field subtraction was utilized and found unsatisfactory for use with signals of 20 msec duration in a reverberant tank. Array processing using a rectangular amplitude function was marginally successful. A Chebyshev amplitude weighted array was successfully used to demonstrate the ability of a linear array to discriminate between the steel plate and free surface scattering.

A computer aided interactive design tool was written to aid in the analysis of array architectures by calculating hydrophone amplitude weights, beamwidth, directivity index and graphically presenting the normalized directional beampattern. The program allows comparative analysis of two array designs simultaneously.

Thesis Supervisor: Dr. R.H. Lyon, Professor of Mechanical Engineering

Thesis Reader: Dr. I. Dyer, Professor of Ocean Engineering

Acknowledgements

I would like to extend my appreciation to Dr Richard H. Lyon who as my thesis advisor has provided guidance and direction throughout the course of this research.

I would like to thank the U.S, Navy for the financial support which allowed me to pursue such an extended period of academic work so far into an established career.

Finally and most importantly I must thank my wife, Cindy, and children, Christopher and Meghan for their continued support and encouragement throughout this period. Their sacfifices during this time at MIT and my preceding ten years of sea duty will remain something for which I will always be indebted.

Table of Contents

Table of Figures

Table of Tables

Nomenclature

$a(t)$	Amplitude of the transmitted periodic signal
c	Sound speed in feet/second
d	Distance between adjacent hydrophones
DI	Directivity index
$F(\Theta)$	Array factor
$G(\Theta)$	Normalized directional gain pattern
I_j	Amplitude weight of j^{th} hydrophone
I_0	Modified Bessel function of order 0
k	Wavenumber
m_x, m_y	Mean of temporal signals x(t), y(t)
M	Number of elements constituting an array
n	Taylor parameter to determine acceptable sidelobe levels
p	Normalized array length ($-1.0 \leq p \leq 1.0$)
P	Pressure
R_{xy}	Cross correlation between x(t) and y(t)
R_{dB}	Desired sidelobe level in dB
$S_R(t)$	Signal at the hydrophone
$S_T(t)$	Signal at the projector
t_p	Pulse duration
z_0	Chebyshev polynomial
α	Pedestal height for cosine amplitude taper function
α_j	Variable used in calculation of amplitude tapered array factor
λ	Wavelength in feet

ω	Carrier frequency in radians/second
$\varphi(t)$	Phase modulation
ρ_{xy}	Correlation coefficient or normalized covariance
σ_x	Variance of temporal signal x(t)
τ	Time delay between signal arriving at adjacent hydrophones
Θ	Angular direction from array axis
Θ_0	Main response axis, "look direction"

1 INTRODUCTION

1.1 Purpose of the Research

The scattering of acoustic energy from fluid loaded structures has been a topic of interest for many years. Recently attention has shifted to considering active control methods to reduce/eliminate the energy reflected from these structures. An active control system would be capable of detecting the incoming wavefront, determine its physical properties, and through an adaptive control system cause the structure to respond in a manner which will reduce the reflection of the incident wavefront.

As these studies mature, the need for inwater experiments becomes necessary to validate the theory and test the control system algorithms. While experiments in open water would be most like the anticipated operating environment they would also be prohibitively expensive. A likely alternative is to conduct the inwater tests in a tank facility. This approach has a problem in that the physical size of the tank lends itself to developing a large number of reflections of the original signal off the sides, bottom, and free surface tending to corrupt that part of the signal which undergoes specular reflection from the fluid loaded structure. The researcher requires quantifying the change in scattered energy from the structure due to the control system in order to rate the performance of a chosen method.

A simple solution to the problem is to arrange the experiment in the tank such that the scattered signal reaches the hydrophone before the second multipath signal. In this way time clipping the signal appropriately will show the reflected signal alone. Figure 1 shows a typical setup for this type of experiment.

The significant drawback to this method is that it artificially requires the active interrogation of the structure to be a short duration pulse in order to assure that the desired scattered signal is discernable in the hydrophone output. As the associated signal

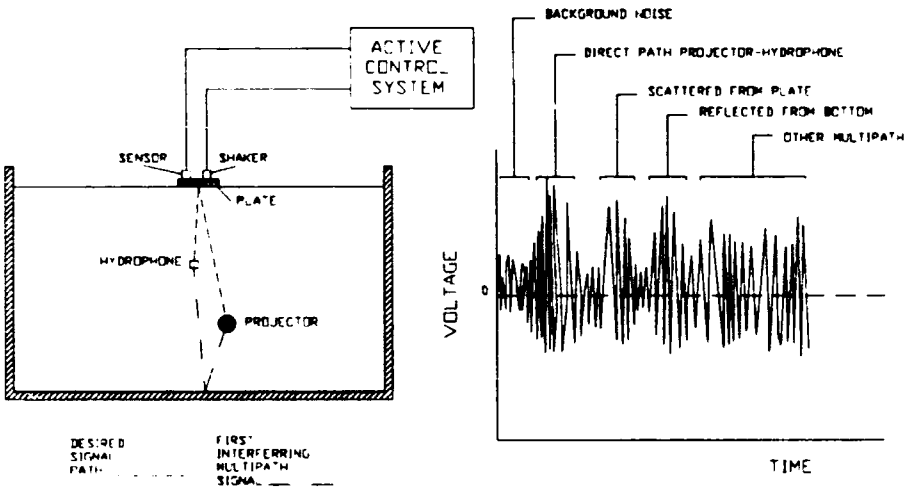


Figure 1.1 Timeclipping to eliminate multipath interference

time history implies, separation of the desired signal from direct path and multipath signals may be difficult. The short active pulse reduces the time available to the control system for detection, determination of the signal characteristics, and initiation of an appropriate control sequence. Alternatively, and the purpose of this report, a series of hydrophones can be used to create an array, a spatial filter used to discriminate signals from directions other than the desired look vector. Figure 1.2 shows the previous experiment with a 9 element array used to spatially filter out the signals from non-look directions. The main lobe of the array is directed at the target of interest while the projector is placed close to a null in the beam pattern minimizing detection of the direct signal.

The use of a tank for the experiment creates an undesirable environment for an array. Multipath signals are numerous and correlated. Phase cancellation of the signal

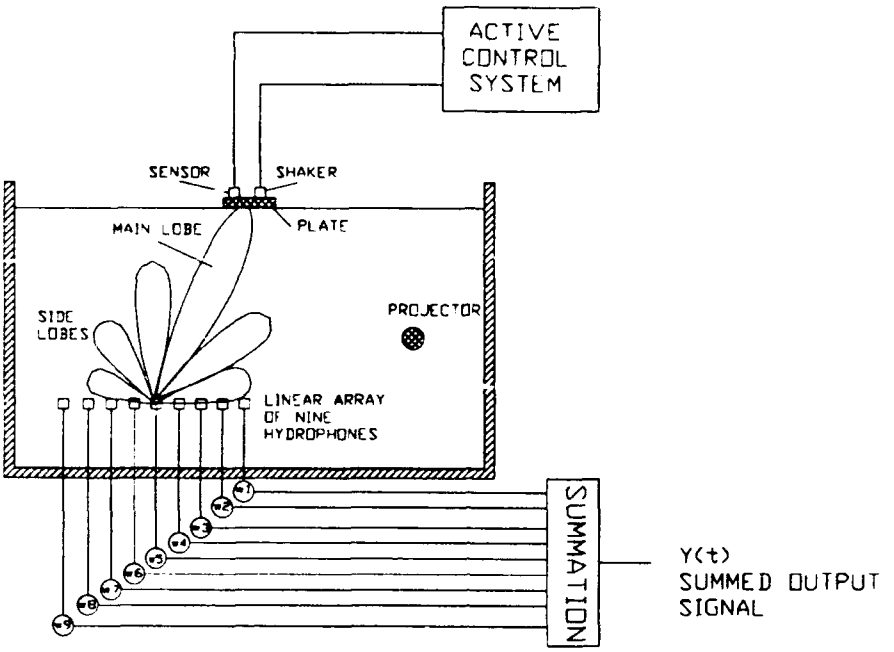


Figure 1.2 Nine element array used as a spatial filter

from the desired direction is a strong possibility. An advantage of array processing is its ability to filter signals from undesired directions allowing longer active interrogation of the structure which in turn gives the control system more time to respond.

1.2 Research Goals

The purpose of this research is to evaluate array processing for use in supporting continued experiments in the active control of scattering. The objective is to determine a beamforming scheme supporting a longer active interrogation of the structure. This in turn gives more response time to the detection and active control system.

As currently envisioned the initial experiment in active scattering will be run using a time clipped short duration pulse of approximately 2 msec to eliminate any multipath

signal detection. The goal of this research is to determine an array beamforming method which will give an order of magnitude increase in the active pulse length. The stated objective is to allow active interrogation of up to 20 msec. This increase should greatly enhance the quantitative assessment of the active control system performance.

A secondary deliverable product as a result of this research is an interactive computer aided design tool which can be used to assist in visualizing the spatial gain due to array processing. The program models the array gain of two separate designs such that a user could easily discriminate between the tradeoffs associated with each. It is intended that this program be used as a first estimation of the characteristic design parameters of a linear discrete element array.

1.3 Organization and Content

The report is divided into four major sections; theoretical analysis experimental setup, computer support, and analysis of results. The first, theoretical analysis, discusses the mathematical development of the beamforming algorithms used as well as an estimate of their applicability to the specifics of beamforming in a reverberant tank. The numerical methods employed in creating an interactive program used to design a multielement line array are presented. The major subheadings include beamforming, phase and amplitude taper, and output power calculations. The second section deals with the experimental setup in the reverberant acoustic facility as well as methods of data collection, reduction, and processing. The third section deals with an interactive array design tool written to assist in determining the capabilities of various array architectures. The final section deals with an analysis of the experimental results as well as recommendations for future study.

2 Theoretical Analysis

2.1 Signal Characterization

The following discussion and throughout this paper the nomenclature of Elliott [15] is used. The transmitted signal, a finite duration pulse in this study, is denoted by $s_T(t)$ at the transmitting element (projector).

$$s_T(t) = \begin{cases} a(t)\cos(\omega t + \phi(t)) & 0 \le t \le t_p \\ 0 \qquad everywhere \quad else \end{cases} \tag{2.1}$$

Where a(t) is the amplitude, ω the carrier frequency, $\varphi(t)$ the phase modulation, and t_p the pulse duration. It is also possible to examine the signal at a receiving element (hydrophone). For a hydrophone located a distance r from the projector the received signal would be of the form:

$$s_R(t) = \frac{1}{r} s_T\left(t - \frac{r}{c}\right) \tag{2.2}$$

Noting that c is the wave propagation speed in water it is obvious that the received signal is proportional to the time delayed projected signal and that this time delay is simply that due to the wave progressing from the projector to the hydrophone. It is common practice to denote the signals represented by equations 2.1 and 2.2 as equivalent complex signals where the actual signal is the real part of the complex representation.

$$\bar{s}_R(t) = \sqrt{P}\, e^{j(\omega t + \phi(t) + \theta)} \tag{2.3}$$

$$= \cos(\omega t + \phi(t) + \theta) + j \sin(\omega t + \phi(t) + \theta) \tag{2.4}$$

Where $s_R(t)$ is equal to the real part of Equations 2.3 and 2.4.

It is often convenient to examine the information carrying part of the signal by eliminating the carrier. The baseband reduced complex envelope, Pillai [13], or complex baseband demodulated version Elliott [15], of a signal eliminates the carrier portion of

the signal leaving a real inphase part and an imaginary quadrature part. Creation of the baseband reduced complex envelope can be accomplished by using a Finite Impulse Response (FIR) linear phase lowpass filter [15].

$$s_R(t) = \sqrt{P}\, e^{j(\phi(t)+\theta)} \tag{2.5}$$

$$= i_{s_R}(t) + j q_{s_R}(t) \tag{2.6}$$

2.2 Statistical Considerations

Acoustic signal processing often relies on a statistical representation of the signals encountered. Relying on the definitions and nomenclature of Newland [5], for two time varying signals x(t) and y(t) with means m_x and m_y and variances σ_x and σ_y respectively, the correlation coefficient or normalized covariance is:

$$\rho_{xy} = \frac{E\{(x - m_x)(y - m_y)\}}{\sigma_x \sigma_y}$$

The value of ρ_{xy} lies between 0 and 1. A value of 1 indicates coherent signals. A value of 0 indicates uncorrelated signals. In fact, most signals lie somewhere between these extremes.

Defining E[f(t)] as the expected value of the function f(t), the cross correlation between two functions of time is defined as:

$$R_{xy}(\tau) = E[x(t)y(t+\tau)]$$

Both cross correlation and correlation coefficient will be referred to throughout this report.

2.3 Conventional Beamforming Analysis

2.3.1 Enhancements Due to Array Processing

Arrays have become common in both radar and sonar applications. The main function of array processing is to create a larger Signal to Noise Ratio (SNR) by using judiciously selected time or phase delays which cause coherent addition of the desired signal and incoherent addition of noise and interfering signals. Figure 2.1 shows a hypothetical source s(t) and a four element linear array of length L and element spacing d'.

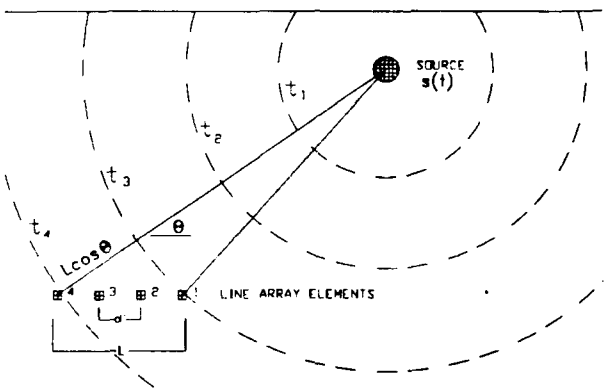


Figure 2.1 Linear array coherent addition

If the source is sufficiently distant from the array we can assume that the energy arrives in the form of an advancing plane wave from a direction Θ with respect to the array axis. It is seen that the wavefront passes array element 1 at time t_3 and element 4 at time t_4. Noting again the assumption of plane wave progression the wave travels a distance of L cos Θ at speed c between element 1 and element 4. Simply stated the signal at element 4 at time t is the same signal that passed element 1 some time (τ) earlier. Letting n refer to the element number and c be the wave celerity:

$$\tau = \frac{L\cos\Theta}{c}$$

$$= \frac{n(d')\cos\Theta}{c} \tag{2.5}$$

If the signal received at hydrophone 1 $s_{R1}(t)$ is known, as well as the source frequency, we can express the expected signal due to the source at each successive hydrophone (n) as $s_{Rn}(t) = s_{R1}(t-\tau)$.

$$s_{Rn}(t) = s_{R1}(t-\tau) \tag{2.6}$$

$$= s_{R1}(t)e^{-j\omega\tau} \tag{2.7}$$

$$= s_{R1}(t)e^{\frac{-j\omega n(d')\cos\theta}{c}} \tag{2.8}$$

$$= s_{R1}(t)e^{\frac{-j2\pi n(d')\cos\theta}{\lambda}} \tag{2.9}$$

By phase shifting, hydrophone outputs the signal from a desired "look direction" can be added coherently along the array. The actual output of any hydrophone in the array is a sum of the signals from all sources as well as other noise. If the noises are considered uncorrelated and the other sources are sufficiently separated from the desired look direction in angular space then Pillai [15] and Steinberg [16] demonstrate that for a single source case with uncorrelated identical element noise, the array SNR is a factor of M greater than single hydrophone element SNR. M is the number of elements in the array.

2.3.2 Distributed Array Normalized Directional Pattern

Figure 2.2 shows a schematic representation of an array beamformer. Each hydrophone element is connected to a phase shifter. The phase shifted signal is then summed over the array to create the output signal z(t). The phase shifters allow the array to be spatially selective as explained in the previous section. Shifting the look direction

changes the weighting factor for each element and effectively steers the array. Continuously changing the weighting factors will cause the array to scan through angular space.

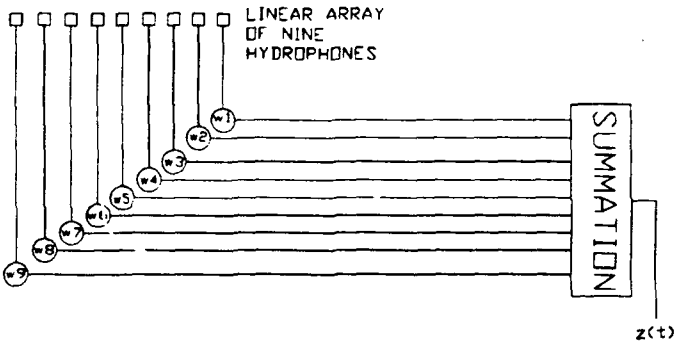


Figure 2.2 Schematic of a Beamformer

The directional pattern can be found by analyzing the array factor defined in equation 2.10 which is simply the summation of the element time shifts over the M elements constituting the array.

$$F(\Theta) = \sum_{j=1}^{M} e^{j\frac{2\pi}{\lambda}d\cos\theta} \tag{2.10}$$

Calculation of the directional pattern can be accomplished by examining the summation of the finite series. Using equation 2.10 with an interelement spacing of d for the 3 element array:

$$F(\Theta) = \sum_{i=1}^{3} e^{j\frac{2\pi}{\lambda}d_i\cos\theta}$$

$$= \sum_{d_i=0}^{2d} \cos\left(\frac{2\pi}{\lambda}d_i\cos\theta\right)$$

$$= 1 + \cos\left(\frac{2\pi}{\lambda}d\cos\theta\right) + \cos\left(2\pi 2\frac{d}{\lambda}\cos\theta\right)$$

Recognizing the periodic nature of cosine functions the second two terms can be combined.

$$F(\Theta) = 1 + 2\cos\left(\frac{2\pi}{\lambda} d\cos\theta\right)$$

The array directional pattern is the result of a finite series the number of terms depending on the number of array elements. Dyer [17] generalizes for an odd number of elements:

$$F(\Theta) = 1 + \frac{2\cos\left(\frac{(M+1)\pi}{2\lambda} d\cos\theta\right)\sin\left(\frac{(M-1)\pi}{2\lambda} d\cos\theta\right)}{\sin\left(\frac{\pi d\cos\theta}{\lambda}\right)}$$

And for an even number of elements:

$$F(\Theta) = 2\frac{\cos\left(\frac{(M)\pi}{2\lambda} d\cos\theta\right)\sin\left(\frac{(M)\pi}{2\lambda} d\cos\theta\right)}{\sin\left(\frac{\pi d\cos\theta}{\lambda}\right)}$$

Defining normalized directional gain pattern.

$$G(\Theta) = \left|\frac{1}{M} F(\Theta)\right|^2 \tag{2.11}$$

Multiple sources, Pillai [13], Widrow [14], Steinberg [16], derive the simplified normalized directional gain pattern for an array of distributed periodic elements. The directional or radiation pattern is the same regardless of even of odd number of array elements and is of the form in equation 2.12. To visualize the beam pattern the value of Θ is varied over the spatial angle of interest (ie 0 to 180 degrees).

$$G(\Theta) = \frac{\sin\left(Mkd\frac{\cos\Theta}{2}\right)}{M\sin\frac{(kd\cos\Theta)}{2}} \tag{2.12}$$

Simplifying Equation 2.12 by substituting $k=2\pi/\lambda$ leaves:

$$G(\Theta) = \frac{\sin\left(M\pi d \frac{\cos\Theta}{\lambda}\right)}{M \sin\frac{(\pi d \cos\Theta)}{\lambda}} \tag{2.13}$$

Steering the beam to a desired look direction is accomplished by adding a constant phase factor to each of the element outputs. This has the effect of shifting the main beam and is incorporated in equation 2.11 by substituting $\cos(\Theta)$ -$\cos(\Theta_0)$ for $\cos(\Theta)$ where Θ_0 is the desired steer angle.

$$G(\Theta) = \frac{\sin\left(M\pi d \frac{\cos\Theta - \cos\Theta_0}{\lambda}\right)}{M \sin\left(\pi d \frac{\cos\Theta - \cos\Theta_0}{\lambda}\right)} \tag{2.14}$$

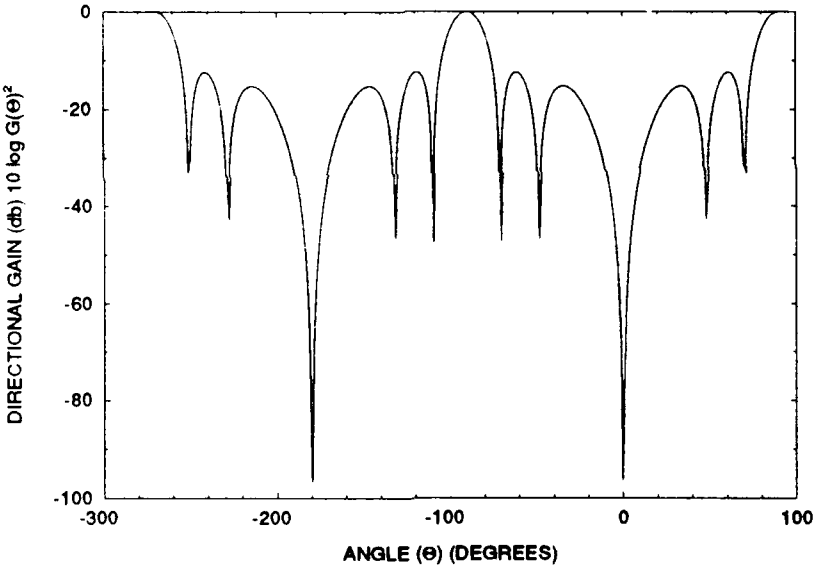


Figure 2.3 Six element array steered to broadside

Equations 2.12 through 2.14 are used to make first approximations as to the ability of an array to meet certain design objectives. Assuming that the physical properties of the signal and the medium (ie frequency, sound speed etc...) are established for the

designer only the features of the array are capable of modification. Limiting the discussion to omnidirectional elements in a linear pattern allows the array designer to vary the number of elements, interelement spacing, and steer direction to create an acceptable array.

Figures 2.3 through 2.7 are plots of normalized directional patterns using equations 2.13 and 2.14. In all cases the array elements are considered omnidirectional hydrophones, sound speed is 4800 feet/second, the signal is a 2 kHz sin wave ($\lambda = 2.4$ feet), and the arrays are linear.

Figure 2.3 shows a six element array steered to broadside, perpendicular to the array axis. The interelement spacing is one half wavelength. The directional pattern is plotted as a log function while the direction along the x-axis is linear. In the coordinate system established in Figure 2.1 the steer direction angle is from the array axis. Other sources, Steinberg [16], Dyer [17], like to define the look angle from broadside, which simply replaces the cosine function in equations 2.12 through 2.14 with a sine function.

In Figure 2.4 the number of elements are increased from six to twelve keeping interelement distance at one half wavelength. Both the 6 element and twelve element arrays are superimposed to show the effect. The side lobes are reduced and the main lobe is narrower while the array is twice the size. It is interesting to note the significant nulls between the sidelobes and the major null at endfire.

Figure 2.5 demonstrates an important point, spatial aliasing of the signal. If the interelement distance is increased to exactly one wavelength, it is easy to show that the signal will now add coherently at the endfire angles where it was previously a null. This results in a spatial gain at end fire equal to that of the mainlobe, definitely not a desired attribute. For this reason interelement spacing should not normally exceed one half wavelength. Keep in mind that for any chosen spacing there is a frequency above which aliasing will occur.

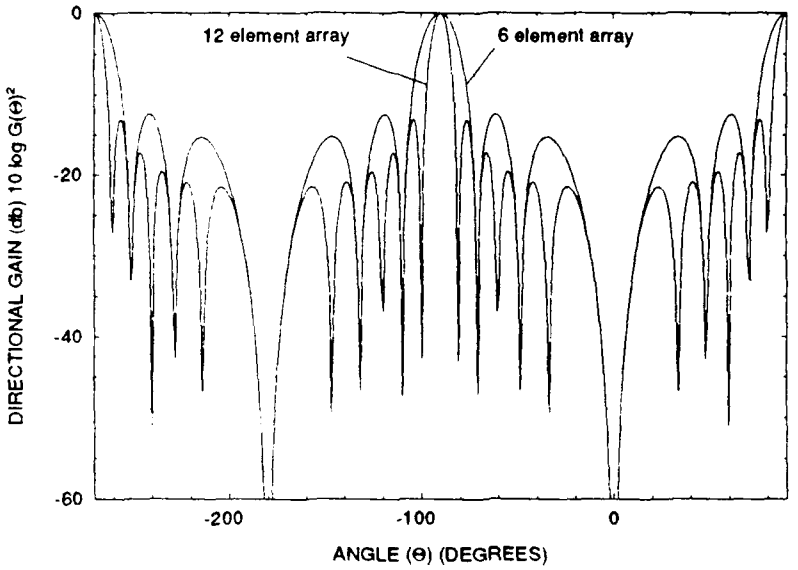


Figure 2.4 Twelve element array steered to broadside

This effect creates grating lobes which significantly reduce the effectiveness of the array to act as a spatial filter. In order to eliminate grating lobes the interelement distance should not exceed $\lambda/2$ for the highest frequency of interest. An alternate visualization is that grating lobes will always exist but judicious selection of interelement separation will force them to occur outside the visible region ($0° \leq \Theta \leq 360°$).

Decreasing the interelement spacing below one half wavelength will widen the beams and increase the sidelobe level. Another simple calculation will verify that a 6 element array with $\lambda/2$ spacing will have the same beam pattern as a 12 element array with a spacing of $\lambda/4$. The difference though is that at any higher frequency of signal the 6 element array will begin to exhibit spatial aliasing while the 12 element array will actually improve in performance.

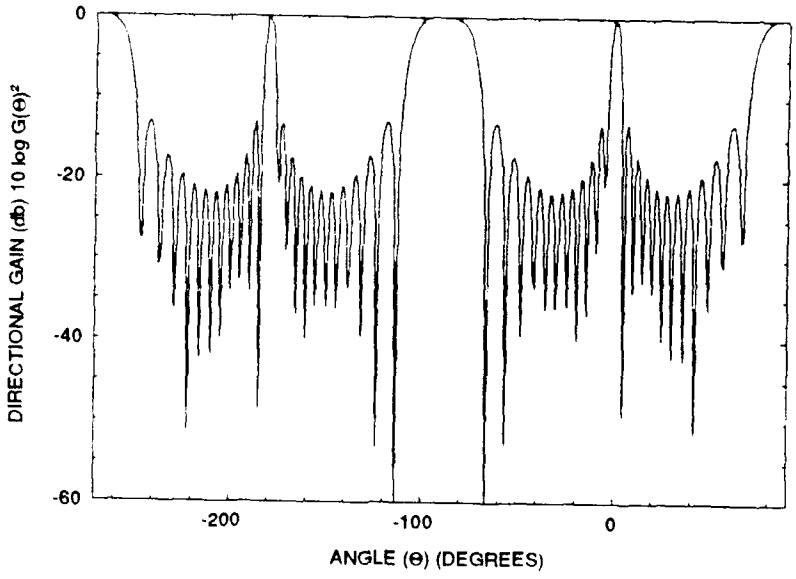


Figure 2.5 The effects of interelement spacing on aliasing.

Figure 2.6 uses equation 2.13 to show how an additional phase shift factor can be used to steer the array from broadside to some other desired look direction. Here the array is steered to an angle of 45 degrees forward of broadside. This figure clearly shows a second main response axis beam at 315°. Linear arrays form beams which are in fact cones symmetric about the array axis. The beampattern plots are two dimensional sections of the three dimensional beams showing both sections of the cone. Correct visualization of the area in physical space that the main beam covers requires rotating the beampattern plots through 180° in angular space. Figure 2.7 is a representation of the 3 dimensional beams formed by a linear array. Beams are shown at endfire, 30°, 60°, and broadside tot the array axis.

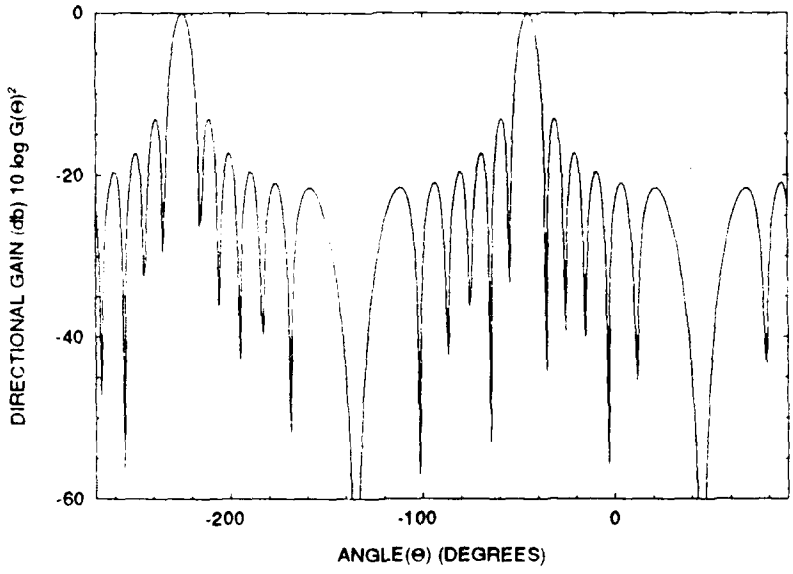


Figure 2.6 Twelve element array steered 45 degrees

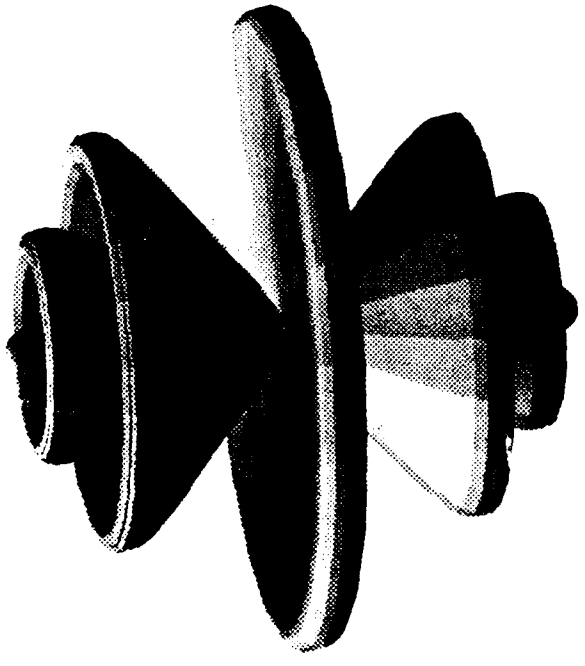


Figure 2.7 Linear array beams in 3-D.

2.3.3 Effects of Array Amplitude Taper

Equation 2.10 which defined the array factor F(Θ) was simplified in that the individual elements were uniformly spaced and evenly weighted with an amplitude taper of unity across the array aperture. A more stringent definition is equation 2.15 where it is possible to individually weight the amplitude of each element's output. Each element now can be phase tapered as well as amplitude tapered to create the most desirable directional gain pattern.

$$F(\Theta) = \sum_{j=1}^{M} I_j e^{j\frac{2\pi}{\lambda} d \cos\Theta} \tag{2.15}$$

In an attempt to quantify the merit of an array the designer relies on three factors. Main lobe beamwidth is calculated at the half power point or alternately at the point where the normalized directional gain, G[θ], is reduced by 3 dB. Sidelobe level, the second factor, is the value of G[θ] at the peak of the first sidelobes. Directivity factor (d) is the expected (average) value of $(F[\theta])^2$ over angular space and is a dimensionless factor. Directivity factor (d) has a value less than 1.0. The lower the value of d the more efficient the array can be considered.

$$d = \frac{\int F(\theta)^2}{4\Pi}$$

The directivity index, DI, is the ratio, in decibels, between the power received by an omnidirectional hydrophone in an isotropic field to the power received by the array beam. The larger the value of DI the more effective the array in filtering signals from non-desired directions.

$$Directivity \quad Index = 10\log\left(\frac{1}{d}\right)$$

$$= 10\log\frac{4\Pi}{\int(F(\theta))^2}$$

Amplitude tapers have several effects associated with their use.

- Increased taper can decrease the associated side lobes, perhaps significantly. This is the single most important positive attribute of amplitude taper.
- Amplitude taper decreases the directivity index a small amount.
- Main lobe beamwidth is increased as is the angular distance between zeros.

The design program ARRAY in Appendix A allows the user to select several typical amplitude taper functions. Alternatively the user can enter specific values of amplitude taper for each element and obtain a graphical representation of the expected beam pattern.

Incorporation of variable amplitude weighting in equation 2.15, eliminates the ability to derive equations 2.13 and 2.14 by successive simplification of the finite series. As before, the array factor $F(\Theta)$ varies depending on an even or odd number of array elements, but unlike the uniformly tapered case, the final result is unique for each case. A generally applicable equation, such as 2.14, is no longer available. Figure 2.8 shows two linear arrays of equally spaced elements steered to an arbitrary direction Θ. The reference position for determining hydrophone location is the center of the array. Array A has an odd number of elements and Array B has an even number of elements.

Balanis [18] solves equation 2.15 for the two cases portrayed in figure 2.7 and arrives at equations 2.16 and 2.17 for the array of even and odd elements respectively. A change in nomenclature occurs here where the array has either 2M or 2M+1 elements where M is an integer value.

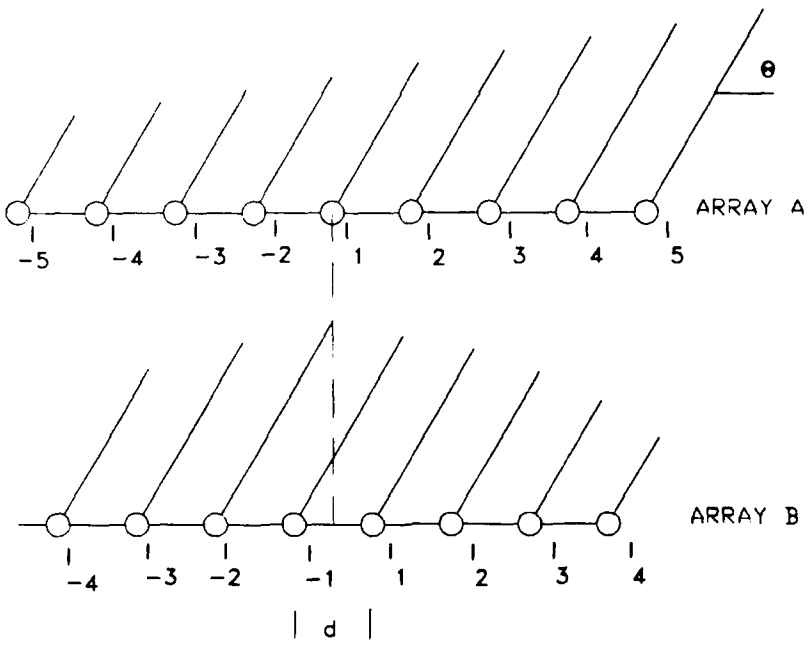


Figure 2.8 Amplitude tapered arrays of even and odd elements

$$F(\Theta)_{2M} = \sum_{j=1}^{M} I_j \cos\left(\frac{(2j-1)}{2} kd\cos(\Theta)\right) \tag{2.16}$$

$$F(\Theta)_{2M+1} = \sum_{j=1}^{M+1} \frac{I_j}{\alpha_j} \cos((j-1)kd\cos(\Theta)) \tag{2.17}$$

It should be noted that α_j is defined as:

$$\alpha_j = \begin{Bmatrix} 2 & j=1 \\ 1 & j \neq 1 \end{Bmatrix}$$

There are a multitude of taper functions applicable to signal processing. The design tool ARRAY written for this research allows the user to interactively select from a menu of six choices. The numerical methods used in incorporating these techniques are explained in detail in Chapter 4. The rectangular, cosine and raised cosine apertures are

based on the presentation by Steinberg [16], Taylor window functions are from Collier [6], Balanis [18] and Pozar [19], and the Chebyshev Array is from Balanis [18] and Pozar [19].

The rectangular aperture usually abbreviated RECT has a uniform excitation over the array length L. This excitation is typically assumed to be unity. The magnitude of the directional gain pattern has a value of -13.5 dB at the first side lobe, the envelope halves with every doubling of spatial angle.

The raised cosine and raised cosine squared taper functions have improved characteristics in that their first sidelobe levels are -23.5 dB for pedestal height of .5 and -32 dB for pedestal height .1 respectively. For a normalized array length p from -1.0 to 1.0 and a pedestal height of α the element excitations are given by equations 2.18 and 2.19. Note that for a selected pedestal height of 1.0 both these taper functions will reduce to a RECT window.

$$I_j = \alpha + (0.5 - \alpha/2)\cos(p\Pi) \tag{2.18}$$

$$I_j = \alpha + (1 - \alpha)\cos^2(p\Pi) \tag{2.19}$$

The Dolph-Chebyshev (Tschebysheff) array will create an array which has the narrowest main lobe in the look direction for a specified side lobe level. A unique characteristic of this array is that all the sidelobes are at the same level. The element excitations are related to Chebyshev polynomials. Several methods of efficiently computing the element weights of the Chebyshev array have been proposed. The method used in ARRAY, a nested product algorithm, proposed by Bresler [21] is recommended by Pozar [19] and Burns [20].

For a desired sidelobe level of R_{dB} in dB, R in volts is

$$R = 10^{(R_{dB}/20)}$$

For an array of M elements the Chebyshev polynomial z_0 is given by equation 2.19.

$$z_0 = \cosh\left(\frac{1}{M-1}\cosh^{-1}R\right) \tag{2.19}$$

Defining the array element numbering scheme as depicted in Figure 2.7, N is the element number furthest from the array centerline. The array element weighting factors are calculated using equation 2.20.

$$I_{N-n} = \begin{cases} (M-1)\alpha NP(n,\alpha) & for \quad n = 1,2,\ldots \mathrm{B} \\ 1.0 & for \quad n = 0 \end{cases} \tag{2.20}$$

$$\mathrm{B} = \begin{cases} N-1 & for \quad M \quad even \\ N & for \quad M \quad odd \end{cases}$$

where:

$$NP(n,\alpha) = \sum_{m=1}^{n} \alpha^{n-m} \prod_{j=m}^{n} f_j^n$$

$$f_j^n = m\frac{(M-1-2n+j)}{(n-j)(n+1-j)}$$

$$\alpha = 1 - \frac{1}{z_0^2}$$

The Taylor weighted aperture, referred to as Taylor Line Source (one parameter), will produce a reduced sidelobe array which unlike a Chebyshev array does not have an optimum main lobe beamwidth to sidelobe level ratio. The array is very close to optimum though. It does require consideration as a design however, because unlike the Chebyshev array which has uniform sidelobe levels, the Taylor array has sidelobe levels which decrease monotonically as the angle increases from the broadside direction. This type of pattern might be advantageous when a known interfering source is located in a direction associated with the third or higher sidelobe. In this case the designer may be willing to trade some small amount of increased main lobe beamwidth for a lower sidelobe in the vicinity of the interfering source, a choice not available to a Chebyshev array. The Chebyshev array has sidelobes at a constant level.

For normalized array length p with values of -1 to 1, Taylor parameter n is a constant input by the designer which relates main lobe width to sidelobe level, and noting that I_0 is a modified Bessel function of order zero, the element excitations are calculated by equation 2.21.

$$I_n = I_0(n\sqrt{1-p^2}) \tag{2.21}$$

Table 2.1, extracted from Balanis [18], documents the expected sidelobe level for the Taylor parameter n.

Table 2.1 Taylor parameter (n) versus first sidelobe level.

	First Sidelobe Level (dB)					
	-15	-20	-25	-30	-35	-40
n	1.1178	2.3204	3.2135	4.0090	4.7551	5.4711

The final window function incorporated into ARRAY is a user defined series of weights. This allows a user not familiar with beamforming theory to experiment with the weight values for each hydrophone position and graphically see the result. Additionally this selection allows weights from an externally calculated analysis to be incorporated into ARRAY where its graphics capabilities can be used to plot the resulting beam patterns. Possible candidates include Hann, Hamming, and Hansen-Woodyard End-Fire Array.

Equations 2.14, 2.16, and 2.17 can now be used to analyze various array architectures including phase and amplitude tapering, by calculating and plotting the directional gain. These, along with the desired amplitude taper functions, calculated by equations 2.18 through 2.21 as applicable, form the main numerical analysis of the design

tool ARRAY discussed in detail in Chapter 4. ARRAY was used to conduct the initial study of array architecture within the constraints of the reverberant tank's physical dimensions as well as the signal limitations imposed by the active control of scattering experiment. The output of the design study is included in Appendix B for review.

2.3.4 Conventional Beamformer Output Power

Often it is convenient to examine the average output power as a function of look direction θ. For array output signal z(t), average power is defined as:

$$P(\Theta) = E[z(t)^2]$$

The summation of the individual hydrophone signals, z(t) has been previously defined for i elements with weights $w_i(\theta)$, and received signal $s_{Ri}(t)$ as:

$$z_i(t) = \sum_{i=1}^{M} w_i(\theta) S_{R_i}(t)$$

Substituting and rearranging into a matr ; formulation the average output power can be rewritten as:

$$P(\theta) = w^T E[z(t)^T z(t)] w$$

For the conventional beamformer the weight applied to each element is the product of a phase taper to steer to a desired look direction, and an amplitude taper to control the sidelobe levels.

$$w_i(\theta) = \frac{I_n}{\sqrt{M}} e^{-j\frac{(2\Pi d)}{\lambda}\cos\theta}$$

2.4 Linear Array Synthesis

Physical constraints imposed by the BBN acoustic tank as well as the array support structure limited the array aperture to approximately 10 feet. The active control of

scattering experiment had been designed to operate at a frequency of 2 kHz. These predetermined constraints allowed the array design to vary hydrophone separation, the positioning of the array , the amount of beam steering or phase taper, and finally the type of amplitude taper. In order to rapidly examine a large number of array architectures the design tool ARRAY was written and used extensively. Figures 2.9 through 2.12 are actual screen images captured while executing ARRAY during the initial design stage.

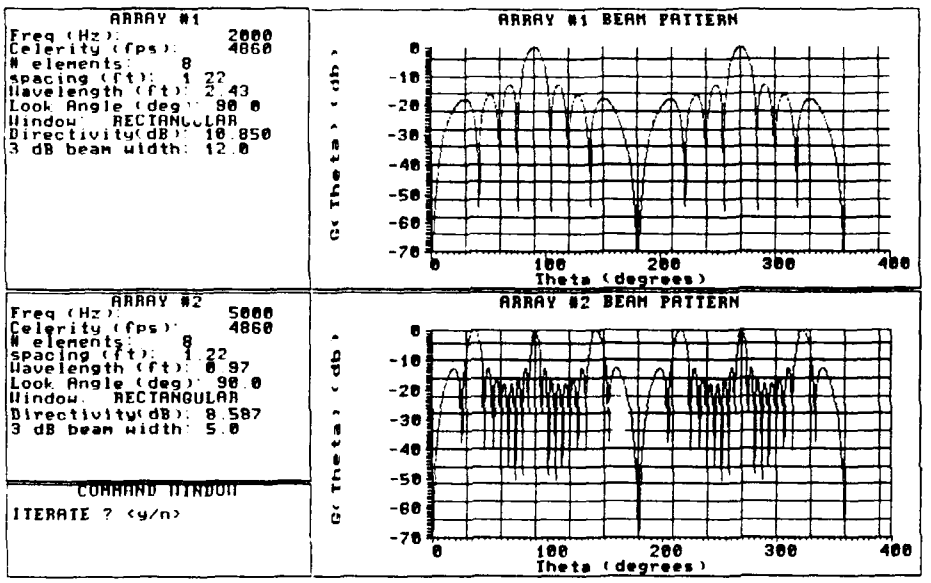


Figure 2.9 Eight element spacing analysis.

Although the active pulse was predetermined to be a 20 msec sine wave at 2kHz it was decided to analyze the acoustic environment following scattering off the actively controlled plate up to 5 kHz. This would allow the effect of the active control system to be better examined over a wider frequency range. At 2kHz, with sound speed of 4860 feet per second (fps), wavelength (λ) is 2.4 feet. Since spatial aliasing is not desired, an interelement distance of 1.2 feet is the maximum allowed. Keeping within the 10 foot aperture allowed results in an 8 element array. Figure 2.9 is the ARRAY screen output

showing the expected beam pattern for this first iteration at both 2 kHz and 5 kHz. For the initial iterations the beam was steered to broadside and a rectangular amplitude taper of unity applied.

As is evident, interelement spacing of $\lambda/2$ at 2 kHz results in dramatic spatial aliasing at 5 kHz and above. The solution requires an analog anti-aliasing low pass filter to eliminate any signal above 5 kHz prior to analog to digital conversion and an interelement spacing which precludes spatial aliasing at the highest expected frequency (5 kHz). Selecting 0.583 feet or 7 inches as an interelement distance, the 10 foot acoustic aperture will support a 15 element array. Figure 2.10 shows the expected beam pattern at the 2 kHz active pulse frequency as well as the highest frequency of interest, 5 kHz. These are considered optimum within the physical constraints imposed.

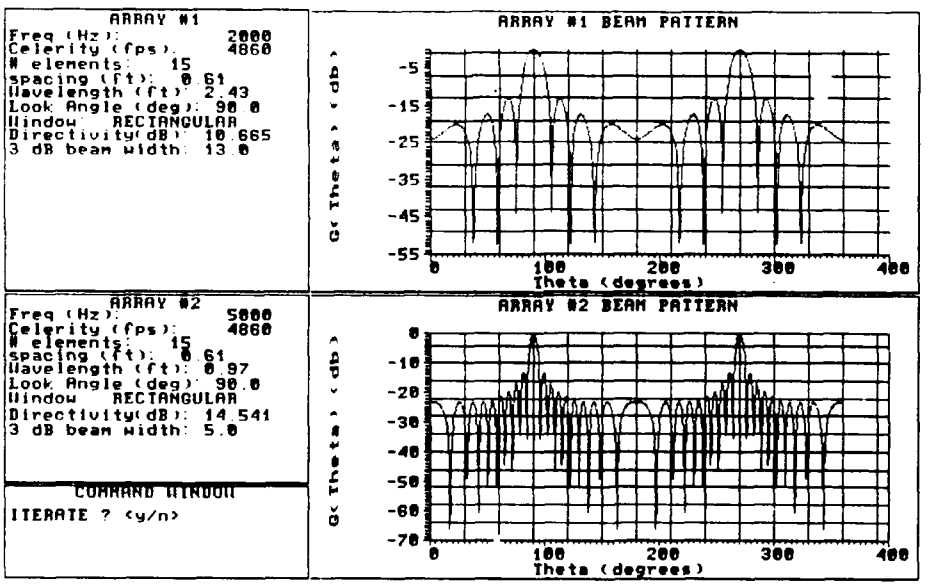


Figure 2.10 Fifteen element array analysis

The design as proposed exhibits higher sidelobe levels than desired. Since the physical design limits preclude adding more elements, it is possible to reduce the sidelobe

levels by employing amplitude taper at the cost of widening the main beam. Figures 2.11 and 2.12 demonstrate the effect of a Taylor window and a Chebyshev window on the 15 element arrays of Figure 2.10. ARRAY was used to determine the element weighting factors for a Taylor parameter of 3 and a Chebyshev sidelobe level of 30 dB. Broadening of the main beam is clearly demonstrated.

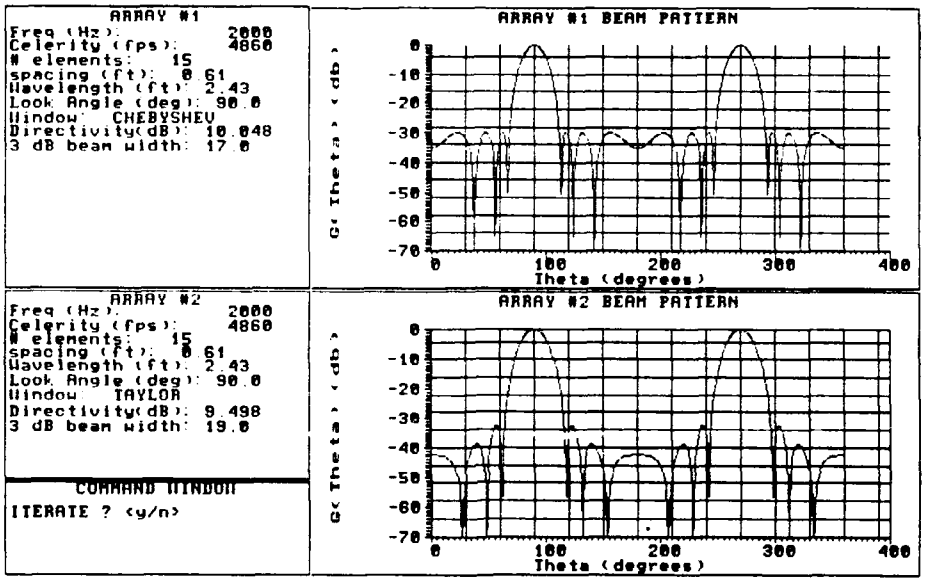


Figure 2.11 Amplitude tapering at 2 kHz.

The element weights calculated by ARRAY for creation of the beam patterns of Figure 2.11 at 2 kHz were manually entered into the 15 element array at 5 KHZ to examine the array response at the higher frequency. This can be noted by the annotation of a USER defined taper window in the array table portion of the figure.

Several other amplitude taper functions were analyzed and the summary of element weights is contained in Table 2.2 along with the value of the highest sidelobe in dB, the main lobe 3dB beam width and the arrays calculated directivity index in decibels. Detailed output, including individual beampattern graphs, tabulated normalized

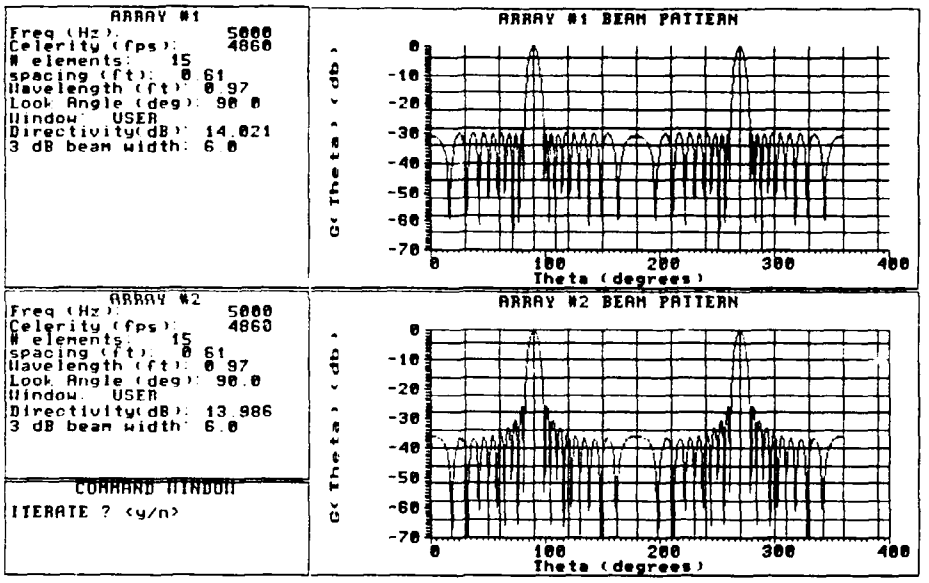


Figure 2.12 Amplitude taper at 5 kHz.

directional gain values and amplitude taper weighting values for the arrays analyzed are included in Appendix B for review as desired. The data included in Appendix B was used to determine the effects on beam patterns due to changes in:

- Number of array elements.
- Frequency response.
- Steering the main response axis.
- Interelement distances.
- Amplitude taper functions.

Appendix B contains example data from examining several iterations of array design. In actuality because of the ease with which a large number of beam patterns can be rapidly analyzed on screen many more types of array architectures were examined and discarded without producing hardcopy.

Table 2.2 Amplitude window weights

	COSINE α=.5	COSINE^2 α=.1	TAYLOR n=3	CHEBYSHEV -30 dB	RECTANGULAR
Element 1	0.2500	0.1000	1.0000	0.2813	1.0000
Element 2	0.2748	0.1446	1.6922	0.3263	1.0000
Element 3	0.3441	0.2694	2.4455	0.4749	1.0000
Element 4	0.4444	0.4499	3.1957	0.6302	1.0000
Element 5	0.5556	0.6501	3.8735	0.7757	1.0000
Element 6	0.6559	0.8306	4.4132	0.8947	1.0000
Element 7	0.7252	0.9554	4.7608	0.9728	1.0000
Element 8	0.3750	0.5000	2.4404	0.5000	1.0000
Element 9	0.7252	0.9554	4.7608	0.9728	1.0000
Element 10	0.6559	0.8306	4.4132	0.8947	1.0000
Element 11	0.5559	0.6501	3.8735	0.7757	1.0000
Element 12	0.4444	0.4499	3.1957	0.6302	1.0000
Element 13	0.3441	0.2694	2.4455	0.4749	1.0000
Element 14	0.2748	0.1446	1.6922	0.3263	1.0000
Element 15	0.2500	0.1000	1.0000	0.2813	1.0000
3 dB Beam	17	21	18	17	14
Sidelobe	-22	-23	-26	-30	-13
Directivity	9.924	9.035	9.788	9.849	10.459

Based on 15 elements spaced 7 inches apart for 2 KHZ signal, c is 4860 ft/sec.

Based on a review of this data the array employed consisted of 15 elements placed 7 inches apart on an array support structure positioned to keep the broadside beam directed at the target plate, which eliminates the need to phase taper in order to steer the main beam to the desired target. The projector would be placed in the null between the first and second sidelobe, approximately 124 degrees off array axis. Post-processing the data would afford the capability of trying several amplitude weighting functions to determine which was best suited to the problem geometry, although a Chebyshev array appears optimum. Due to the fact that this array is specifically intended for spatial filtering of

multipath interfering signals and not extremely accurate direction finding, it is logical to use amplitude tapering and accept a wider main lobe in order to obtain the better reduction in sidelobe levels.

3 Experimental Setup

3.1 BBN Reverberant Tank

In water testing involved forming the synthetic aperture by successive hydrophone placement in the acoustic test facility at Bolt, Beranek, and Newman (BBN) located in Cambridge, Massachusetts. The tank facility is designed for measurement of acoustic power levels. The side walls and bottom of the tank were constructed so as to eliminate parallel opposing walls which could lead to standing wave formation. Figure 3.1 shows general physical dimensions of the facility.

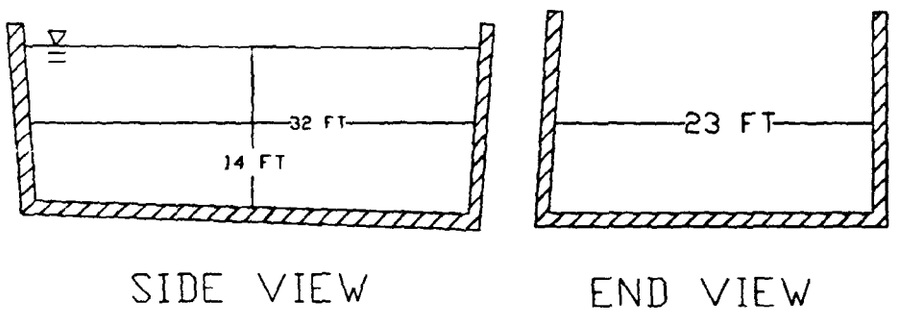


Figure 3.1 BBN Reverberant Water Tank

The reverberant tank holds approximately 11,000 ft^3 of fresh water. Sound speed is estimated to be 4860 ft/sec based on density and temperature considerations. The tank bottom and four walls are lined with a highly reflective polyurethane foam which acts as a pressure release surface. The average absorption coefficient has been previously determined to be 0.05. The tank surfaces being highly reflective tend to create a large number of multipath signals, highly correlated and perhaps coherent, to the original projector pulse. These multipath signals are the predominate problem associated with beamforming and array processing in a reverberant tank.

3.2 Scattering Plate

The target for the acoustic pulse is expected to be a steel plate with an attached control system which will cause the plate to respond as an extension of the free surface. Figure 3.2 shows the physical dimensions of the target plate.

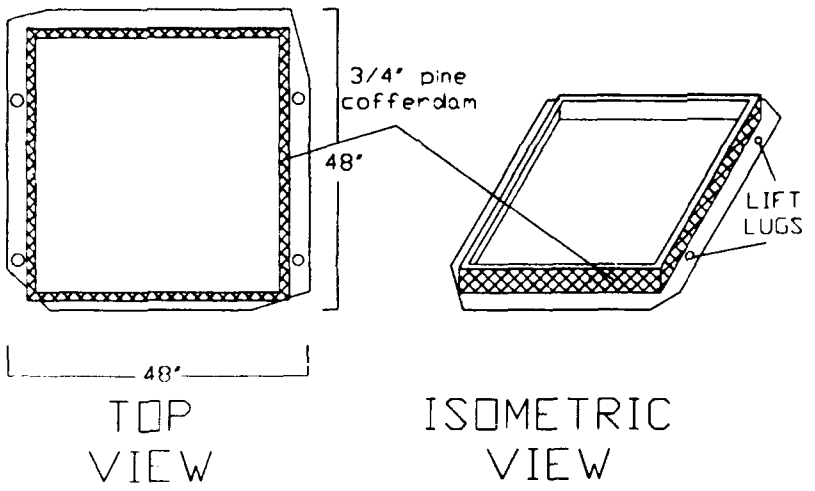


Figure 3.2 Scattering Target Plate

The plate is approximately 16 ft^2 of 1/4 inch steel. The plate has applied damping treatments to establish a damping loss factor η of 1 to 2 percent. In order to allow the plate to be free floating, and to fluid load only one side, a cofferdam of 3/4 inch pine was built. The cofferdam was rigidly attached to the plate and sealed to prevent water entry. The plate, when waterborne, floated with a waterline of 3.5 inches as expected.

3.3 Projector and Active Pulse

The equipment used to generate the active pulse is depicted in Figure 3.3. The pulse consisted of a 40 cycles (20 msec) of 2 kHz sine wave with an amplitude of 4.5 volts and an initial positive slope. A Hewelett Packard signal generator model HP 3314A was used to create the initial signal. This signal splits at the signal generator output with

one path being monitored on a dual trace oscilloscope and recorded as the trigger pulse on channel 3 of a TEAC XR-5000 14 channel tape deck. This signal will be the time synchronization for analog to digital conversion discussed under data collection and reduction.

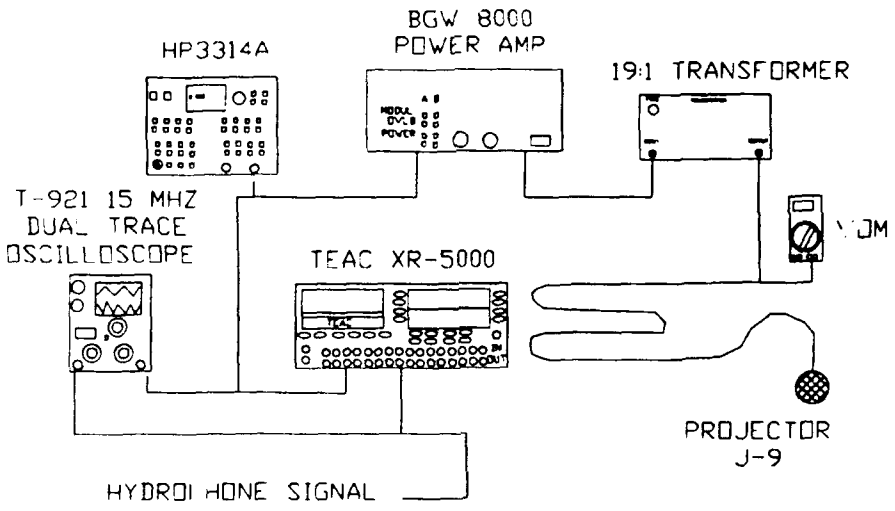


Figure 3.3 Projector schematic

The second signal from the signal generator passes through a single channel of a BGW 8000 power amplifier and a 19:1 transformer. The output of the transformer is monitored by a Beckman VOM to set the voltage to 150 volts using the power amplifier controls. The transformer output is sent to a Sumner & Mills model J-9 4 inch projector. The active pulse can be triggered manually or by time synch selected on the HP 3314. During the experiment the active pulse was automatically triggered at 10 second intervals.

3.4 Hydrophone Assembly

The receiving section of the apparatus is shown in figure 3.4. One Celesco LC-10 cylindrical hydrophone is positioned at 15 separate positions on the array support structure.

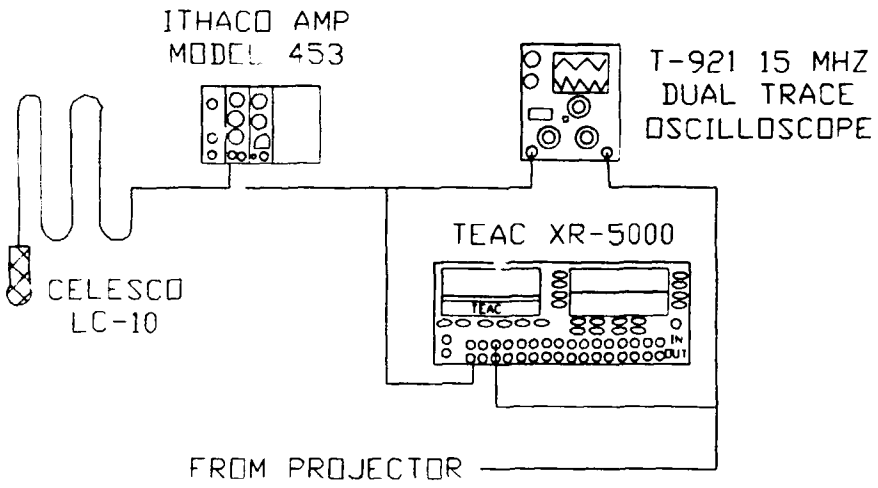


Figure 3.4 Hydrophone schematic

The hydrophone has an omnidirectional response orthogonal to its longitudinal axis and a cartoid response parallel to the longitudinal axis. The hydrophone has a response of -101 dB re 1volt/Pa. The signal is amplified by an ITHACO model 453 amplifier whose signal was monitored using headphones to detect any clipping. The amplifier was set to get an output amplitude close to ± 5 volts as monitored on the dual trace oscilloscope. The signal was recorded on channel one of the TEAC XR-5000 multi-channel tape recorder. Forty pulses were recorded for each of fifteen hydrophone positions. Twenty were taken with the target plate in the water and twenty with the plate removed from the tank facility. A continuous voice track was recorded to facilitate digitization and data transfer at MIT from the TEAC recorder to the MASSCOMP computer.

3.5 Forming a Synthetic Aperture by Hydrophone Position

The acoustic aperture of the array was formed by sequentially positioning the same hydrophone at fifteen separate positions in the reverberant tank. In order to post process the data as if one 15 element array was used required reproduction of the active interrogation pulse and the capability to position the hydrophone accurately. The HP 3314 signal generator was capable of creating repetitive pulses. In order to ensure accurate hydrophone positioning an array support structure was built. Figure 3.4 depicts the experiment as it was placed in the BBN tank. The projector, plate, hydrophone, and array support structure are shown.

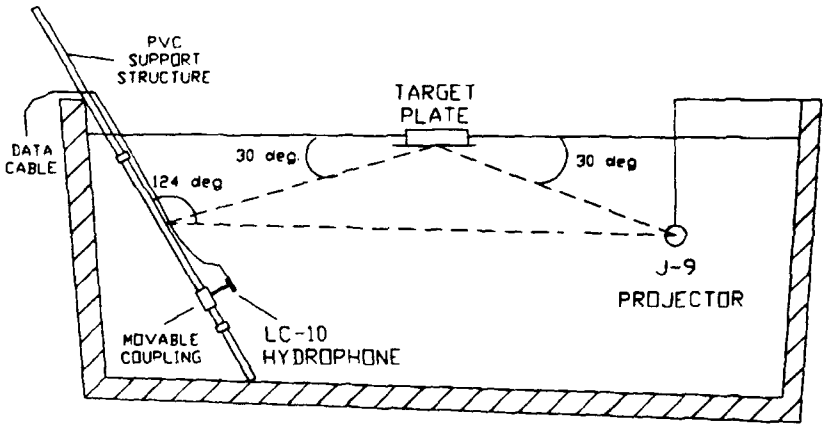


Figure 3.5 Synthetic aperture array structure

The array support structure is 20 feet of 2 inch radius PVC pipe free flooding with water. The pipe is divided into three sections, a 3 foot stub tube at the bottom, 10 feet of acoustic aperture, and 7 feet at the top for handling and support. On the 10 foot acoustic window portion, a 12 inch long, 2.5 inch radius movable collar is mounted. This collar can be positioned by downhaul and uphaul cables to position it along the 2 inch PVC pipe. To prevent azimuthal rotation of the movable collar it is slotted and moves along a track attached to the back of the support structure. A mechanical scale allows positioning

the collar to within 1/16 inch along the array support structure axis. One foot of 3/4 inch diameter freeflooding PVC pipe is rigidly mounted to the movable collar orthogonal to the array axis. The LC-10 hydrophone is attached to the end of this pipe to keep it away from the support structure.

The entire array support structure is positioned in the tank so as to make a 60 degree angle between array axis and tank bottom. This "array steering" allows the broadside beam to be directed toward the target plate and eliminates the need to phase taper for beam steering.

By simple geometry the plate and projector are positioned such that the signal arrives from 30 degrees below the horizontal axis and is reflected at the same angle in accordance with Snell's law toward array window centerline. The plate center is 14 feet 4 inches from the wall near the array. The projector is 10 feet 6 inches from plate center and 7 feet from the tank wall. The projector is placed at a depth of 6 feet 2 inches to establish an angle of 30 degrees with the target plate and to place the projector at 124 degrees referenced to the array longitudinal axis. Referring to Figure 2.9 and the tabulated values in Appendix B shows this should place the direct path signal from the projector in a null.

3.6 Data Collection Reduction and Processing

The hydrophone output data is collected at the BBN acoustic tank and recorded on the TEAC XR-5000 tape deck. Figure 3.5 depicts the digitization of the data at MIT using the MASSCOMP computer. The taped hydrophone data is replayed through a 5 kHz lowpass anti-aliasing filter. The AD12FA analog/digital (A/D) converter along with a SH16FA sample and hold module are used to digitize data. Two analog to digital channels are opened. Channel 0 contains the trigger signal originally recorded as output

from the signal generator. Channel one contains the hydrophone output.

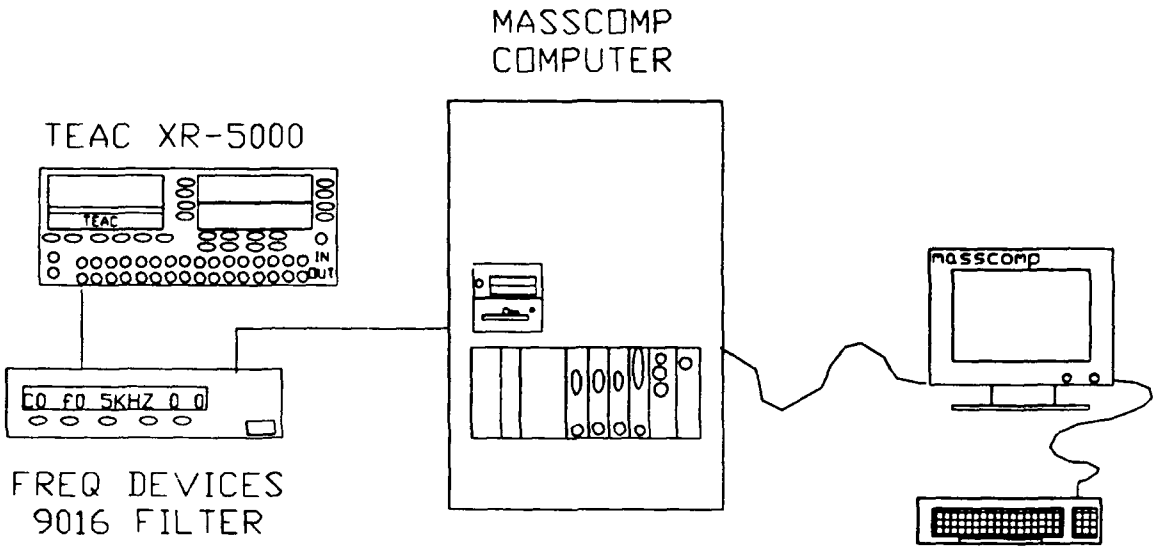


Figure 3.6 Processing of taped data

The data acquisition on the MASSCOMP computer takes place inside the LAB Workbench (LWB) environment. The analog input is demodulated to separate channel 0 from channel 1. The channel 0 signal provides the input for a trigger module which will pass channel 1 (hydrophone) data only after the 1 volt rising trigger is detected for greater than 1 μsec. The output of the trigger module is a time clipped section of digitized hydrophone data which is recorded in a user defined file. The digitized, clipped data is viewed on a digital scope display on the LWB screen. The time window selected runs from trigger detect to 60 msec. This is sufficient time to see the direct path signal as well as plate scattered signal and some interfering multipath signals.

Each individual pulse is digitized, timeclipped and recorded in a file on the MASSCOMP. Figure 3.7 shows the virtual instrument created in the Lab Workbench environment used to digitize, demodulate, timeclip and record a single hydrophone pulse.

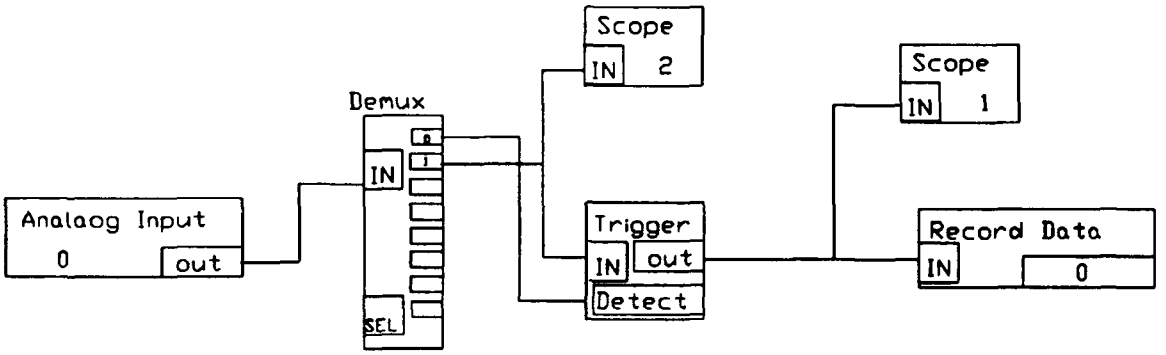


Figure 3.7 Lab Workbench A/D conversion virtual instrument.

Appendix C1 contains a series of plots forming a library of the digitized hydrophone output signals recorded by this method.

4 Programs to Assist in Array Design

4.1 Beamforming Analysis (ARRAY)

ARRAY was written as a design tool to allow rapid iteration of array parameters and to present a graphical view of the resulting beam patterns. Two separate array architectures can be shown on the screen simultaneously to aid in this analysis. The program is intended to be used in the initial phase of array design where it is beneficial to examine many possibilities quickly so as to eliminate unsatisfactory designs before significant effort is expended in numerical analysis or array construction. ARRAY may also prove valuable for those cases where significant postprocessing of prerecorded data is done. ARRAY as currently written will determine the beampattern over 360 degrees in one plane for a linear array of equally spaced elements. The program incorporates phase taper to support beam steering as well as amplitude taper to control sidelobe levels. The code has the capability to be upgraded to plane arrays or non-uniform element spacing. The graphics routines used have the ability to plot 3-D beam patterns should this enhancement prove desirable in the future.

The user has the option to enter the number of elements, signal frequency, interelement distance, sound speed, steer angle and amplitude taper window. The program has been compiled and run on a Video Graphics Adapter (VGA), an Enhanced Graphics Adapter (EGA) as well as Hercules Monochrome Graphics Adapter.

4.1.1 Program Compilation and Use

The source code for ARRAY is included in Appendix A. The program was written in Microsoft Quick C and compiled on an IBM PS/2 Model 70 386 personal computer. The code makes use of the Microsoft extensions to K&R C specifically with respect to

string handling and graphics, and therefore may require modification prior to being ported to non-DOS systems. The program was compiled using the medium memory model with a stack size of 16K bytes in release mode.

The graphics routines used for plotting are modified Microsoft graphics drivers provided by Quinn-Curtis Software for Industry, Science and Engineering as source code. As such the source code is proprietary and not included in Appendix A. The final program ARRAY is based on linking five separate modules together, array.c, menu.c, beam.c, plot.c, and output.c. Array.c is the main calling program from which execution starts. Menu.c is the prime executive which controls initialization as well as program execution. Calls to beam.c, plot.c and output.c are made from menu.c. Beam.c is the primary calculation module where the normalized array gains are calculated as well as the individual element weights based on a selected amplitude taper function. Plot.c controls the graphical user interface, establishing windows, plotting the array gains and updating the array data tables. Output.c opens output files for data recording and ports graphics to the printer as desired by the user.

The program is executed by entering ARRAY at the DOS prompt. Internal to the main calling program ARRAY looks for the file array.con on the default drive. This file contains information regarding the type of graphics driver and monitor installed as well as the printer used. If array.con is found the program loads the hardware configuration data and shifts execution to the program executive, menu.c. Should array.con not be found as would be the case on the first use of the program, a warning is issued and the required configuration data is prompted via menu. The configuration file array.con is opened on the default drive, the data written and the file closed. Program execution continues as before. The file array.con is now available the next time ARRAY is used. To change the hardware configuration delete array.con prior to executing ARRAY. The Hercules version will first load the graphics driver MSHERC and then execute. The

initialization routine in Menu.c assigns values to both array #1 and array #2, calculates array gain and plots the graphs. Figure 4.1 is the first screen displayed and shows the initialization array values. Refer to Figures 2.8 through 2.11 for other examples of the actual screen displays while running ARRAY.

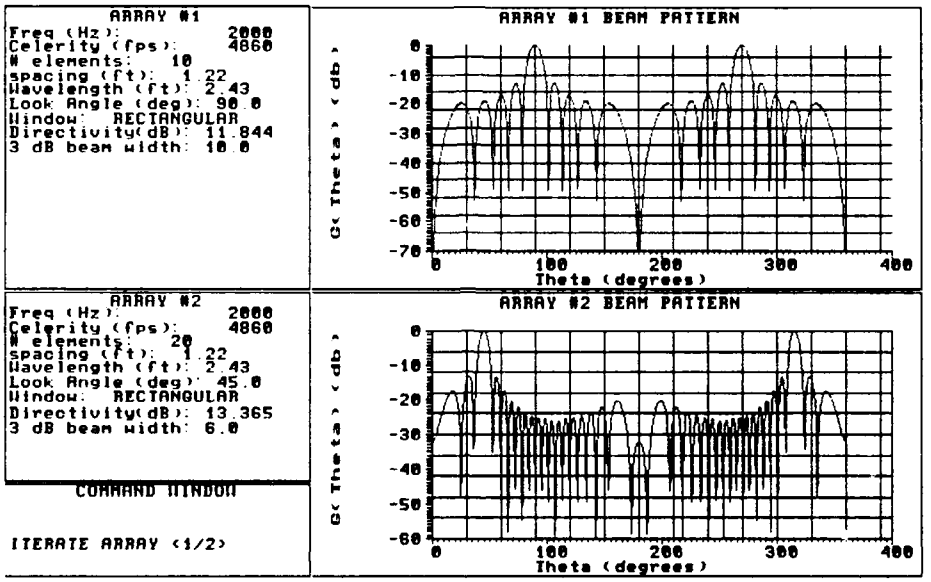


Figure 4.1 ARRAY initialization screen

The command window in the lower left of the screen prompts as to the desirability of iteration. Entering a character [N] will exit the program. Entering a [Y] will cause the command window to prompt for which array to iterate. The user should enter the desired array number([1] or [2]) to be prompted for array parameters. The program terminates the graphics window format for rapid entry of array characteristics. The input screen shows the current default value of each parameter in brackets < > which can be selected by simply using [<- ENTER] vice typing a value. Figure 4.2 shows the parameter entry screen following entry of the sound speed.

During entry a menu of window choices is given. The user should select the table value ([1] through [6]) of the desired choice. After entering sound speed, or accepting the

default, if the window function is any but a rectangular window the user will be prompted for more information. Once the element amplitude weights have been calculated they are presented on screen for verification. The user has a choice of accepting these values or returning to a prior level of data entry depending on the window.

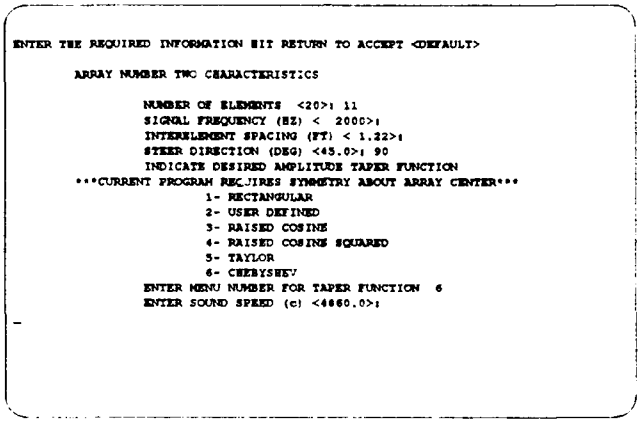


Figure 4.2 Array data entry window.

To accept the element weights as shown enter [Y] and the program returns to a graphics window environment showing the updated array normalized directional gain pattern as well as the new array data table now updated to show the revised array characteristics, amplitude taper selected and the computed values of mainlobe beamwidth and directivity index. Since the program displays two completely independent arrays the user can shift between each at random making iterative changes and examining the results.

The command window again prompts for iteration and the program continues in this manner until no further iterations are desired. When [N] is entered to stop iteration of array designs the command window requests the users desire to produce output from the array synthesis. The user has the option to respond yes, [Y] or no, [N]. Yes will cause the command window to prompt for which array to use 1,2, or both.

Output from ARRAY is directed to two separate devices. The printer declared in the configuration file array.con is used to produce a full page landscape orientation of a single array characteristic table and the associated graph of normalized array gain. Simultaneously, the program opens a file array.dat on the default drive and saves the array characteristics (ie frequency, number of elements etc...), the calculated element amplitude weighting factors, and the tabulated array gain G[θ] at whole degrees between 0° and 180 °. This file is then available for future reference and/or printing as the user desires.

After producing the desired output ARRAY returns to the windows environment. The command window questions if the user desires to quit. A no will return the program to array iteration. A yes will cause program execution to terminate and a return to the operating system.

4.1.2 Numerical Analysis

Three numerical algorithms are employed in calculating the normalized array gain in ARRAY. If the window function chosen is the rectangular function it is numerically much more efficient to use equation 2.14 vice the summations in equations 2.16 or 2.17. Additionally, while not requiring recursive summation loops, equation 2.14 is applicable to either even or odd numbered arrays while equations 2.16 and 2.17 require determining if the array is even or odd numbered. However, if any window function other than a rectangular one is selected only equations 2.15 or 2.16 are applicable. ARRAY will chose equation 2.14 to calculate array gain if the RECT window is selected and either equation 2.16 or 2.17 as required for all other amplitude taper choices.

Equations 2.18 through 2.21 are coded in beam.c to calculate the element window weights. For continuous functions the function value is sampled at the normalized aperture position, from -1.0 to 1.0. This value is assigned to the element weight.

The Taylor window function discussed in Chapter 2 requires the value of a modified Bessel function of order zero. ARRAY incorporates the modified Bessel function algorithm discussed by Press [7]. The function is based on the coefficients of Abramowitz [22] and is included in beam.c as function bessi0( ) for review.

The Chebyshev array element weighting factors are given by Bresler's [21] nested product algorithm in equation 2.20. In order to obtain the correct weighting factors requires solving equation 2.19 for the Chebyshev polynomial z_0. The compiler used to create the executable version of ARRAY does not support hyperbolic and inverse hyperbolic functions. These functions were approximated by their exponential forms as found in Beyer [8]. Exponential and log functions are supported by Microsoft Quick C.

$$\cosh(x) = \frac{1}{2}(e^{x} + e^{-x}) \tag{4.1}$$

$$\cosh^{-1} x = \log_e(x + \sqrt{x^2 - 1}) \tag{4.2}$$

Directivity and directivity index, discussed in Section 2.3.3, are one quantitative assessment of the effectiveness or an array. To calculate directivity requires integration of the normalized directional beam pattern over 360 degrees. ARRAY calculates array response in a plane vice over all physical space. Numerical integration takes place in beam.c during the call to function dir( ). The extended Simpson's rule discussed by Press [7] was used to numerically integrate G[θ] using the 720 calculated values over 360 degrees.

$$\int_{x_1}^{x_N} f(x)dx = h\left(\frac{1}{3}f_1 + \frac{4}{3}f_2 + \frac{2}{3}f_3 + \frac{4}{3}f_4 + \ldots \frac{2}{3}f_{N-2} + \frac{4}{3}f_{N-1} + \frac{1}{3}f_N\right) + O\left(\frac{1}{N^4}\right) \tag{4.2}$$

N is the number of points used in the numerical technique (720), h is the spacing between these points (.5 degrees), f_n the value of normalized directional gain pattern G[θ] at point n, the limits of integration are from 0 to 360 degrees. This technique is a closed formula as the boundary values are evaluated. The formulation of Equation 4.3 in code results in summing overlapping groups of three values. Equation 4.4 demonstrates the technique.

$$\int_{x_1}^{x_N} f(x)dx = h\{\frac{1}{3}f_1 + \frac{4}{3}f_2 + \frac{1}{3}f_3 + \frac{1}{3}f_3 + \frac{4}{3}f_4 + \frac{1}{3}f_5 + \frac{1}{3}f_5 + \frac{4}{3}f_6 + \frac{1}{3}f_7 \ldots \quad (4.4)$$

This gives the required 2/3, 4/3 alternation throughout the interior of the summation equation while preserving the 1/3 factor on the boundary elements. The error expected in Equation 4.3 is on the order of $1/N^4$.

Mainlobe beamwidth is quantified by the -3 dB beamwidth. The values of normalized directional gain are examined at 1/2 degree increments from the desired look direction (where G[θ] = 1.0). At the point where G[θ] is greater than -3dB the iteration ceases. A linear interpolation between the two points bounding the -3dB level is made. The actual beamwidth is twice this value as the beam extends in both directions away from the main response axis (MRA). While not exact, this method is accurate to within fractions of a degree and is certainly adequate for the type of initial design review for which the model was created.

The normalized directional gain pattern is calculated from 0 to 180 degrees and recognizing the symmetry of a line array beampattern these values are simply reproduced at angles 181 to 359. The gain is calculated at half degree values . This appeared to give sufficient resolution to the output plots and improved the accuracy of the numerical

integration required for directivity calculations. Should a higher resolution be desired at a later time the angular step size could be decreased by a simple change to the source code.

ARRAY uses dynamic memory allocation to create arrays and vectors of user defined length at the time of program execution. The normal method of using arrays and vectors in C is avoided as it would place unnecessary limitations on the program or use significantly greater memory than required for the majority of arrays analyzed. An excellent discussion of the rational for this choice is contained in Press [7].

4.1.3 Software verification

Any computer model no matter how sophisticated looking is only as good as the results it calculates. As a check of the coding of the program the same problem could be executed using three separate paths through the code. A RECT window function which uses equation 2.14 should give the exact same results as a USER window function with element weights 1.0, which uses equation 2.16 or 2.17. Similarly, a COSINE on a pedestal window function with the fractional pedestal height α set to 1.0 returns a set of weights equal to the RECT function. Figure 4.2 is the ARRAY output which compares the RECT window with a USER defined window of magnitude 1.0. This exercise demonstrates that ARRAY using equation 2.14 or equation 2.16 will arrive at the same solution. The same comparison was made using COSINE window function with a pedestal height of 1.0.

Function calls to bessi0() and dir() in beam.c calculate the modified Bessel function of order 0 and the integration of the curve of array factor by Simpson's 3/8 method. Both these functions were verified by comparison to known solutions to simplified problems prior to incorporation in ARRAY.

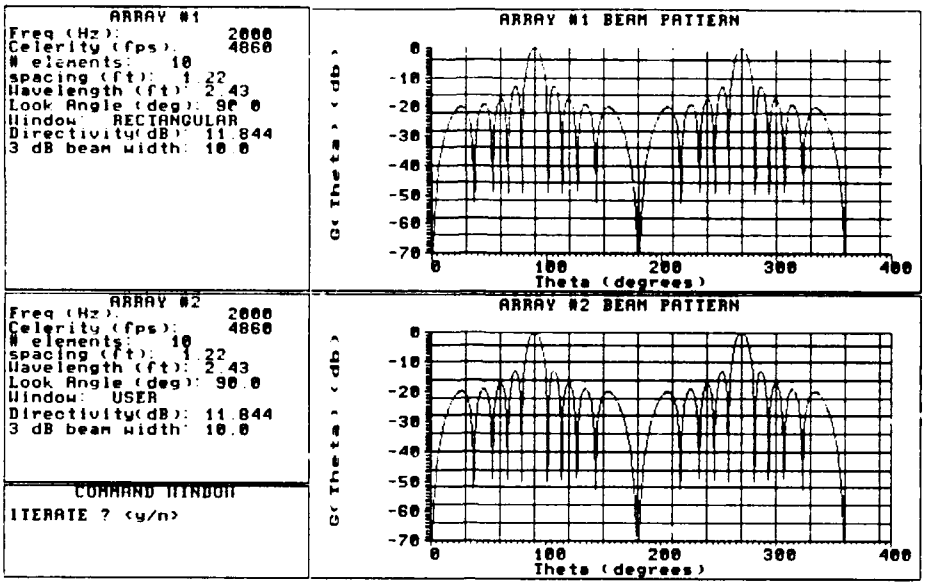


Figure 4.3 Comparison of array gain formulation.

In order to verify the performance of ARRAY, the program was used to verify solutions to array problems found in Collier [6], Pillai [13], Balanis [18], and Pozar [19]. The problems were chosen so as to check as many of the functions of ARRAY as possible. Table 4.2 is a list of some problems run as well as the source of the results used to validate ARRAY as a design tool.

ARRAY was used to compute the amplitude taper element weights, the sidelobe level in dB, the directivity in dB, and the beamwidth of the main lobe. These values were compared to the source document analytical solutions where possible and the error noted. Sidelobe, beamwidth and directivity error are presented as absolute values. Because the element weights varied considerably the largest absolute error for any single element was converted to a percent error. Table 4.3 presents this information.

Table 4.1 Array validation exercises

	PROBLEM STATEMENT	SOURCE DOCUMENT	FUNCTION VALIDATED
1	10 element, d=λ/4, RECT, steer 90 °	[18]	RECT
2	10 element, d=λ/4, RECT, steer 60 °	[18]	RECT, Steer
3	17 element,	[6]	RECT
4	10 element, d=λ/2, USER, steer 90 °, (Binomial element weights)	[18]	USER window
5	10 element, d=λ/2, sidelobe level -26 dB, CHEBYSHEV, steer 90 °	[19]	CHEBYSHEV
6	19 element d=.7λ, n=6, TAYLOR, steer 90 °	[19]	TAYLOR
7	17 element, d=λ/4, sidelobe level -30 dB, CHEBYSHEV, steer 90 °	[18]	CHEBYSHEV
8	17 element, d=λ/4, n= 5.4227, TAYLOR, steer 90 °	[18]	TAYLOR

Table 4.2 ARRAY output error

Problem	Weight Error (%)	Sidelobe Level error (dB)	Main lobe Beamwidth Error(°)	Directivity Error (dB)
1	0*	0.0	0.2°	1.2
2	0*	0.0	0.3°	NC
3	0*	0.25	0.0°	1.81
4	NA	0.0	NC	NC
5	<0.1%	0.0	1.0°	1.6
6	0.1%	0.0	0.1°	1.8
7	0.1%	0.0	NC	NC
8	0.4%	0.0	NC	NC

***- 1.0 across window; NC- not calculated in source; NA-not applicable**

The numerical techniques used in ARRAY appear to be sufficiently accurate. The errors associated with amplitude taper weights, sidelobe level, and mainlobe beamwidth are extremely small. The errors noted in directivity are not unexpected. ARRAY

numerically integrates the area under the normalized directional gain graph and ratios the value obtained to a unity gain level over 360°. The source documents rely on algorithms which calculate directivity for a broadside array only. ARRAY is more robust in that it will calculate directivity irrespective of the look direction. The difference in directivity is not large and the program demonstrates the correct trends in directivity (ie amplitude taper decreases the directivity index a small amount).

5 Signal Processing

5.1 Data Collation and Reduction

Section 3.6 discussed the method of digitizing each pulse by using Lab Workbench on the MassComp computer. Figure 3.7 schematically showed the virtual instrument used. Each pulse of 60 msec duration was stored in an individual file. Each file contained 600 sample points which defined the pulse sufficiently to prevent aliasing up to 5 kHz. For each of the 15 hydrophone positions which defined the synthetic aperture, 40 pulses were recorded, 20 with the scattering plate in position and 20 with the plate removed from the tank.

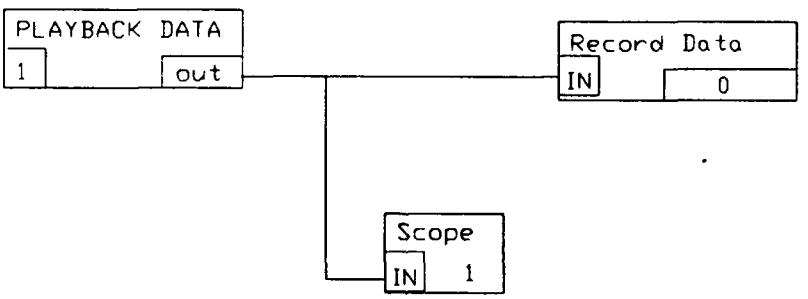


Figure 5.1 Data reformat virtual instrument

Each file was stored as a binary file on the MassComp. A second set of files stored in ASCII format were created to allow transfer from the UNIX based MassComp to a DOS based system. This allowed the raw data to be manipulated on a PC system and eased incorporation into documents. Figure 5.1 shows the virtual instrument used in Lab Workbench to reformat a data file from binary to ASCII prior to data transfer.

The ASCII files created were ported to a NEC Powermate 386 PC based system via a serial port null modem connector using BITCOM communication software. The DOS

system files were then available for data manipulation and tabulation. Each set of 20 pulses specific to a hydrophone position were averaged to obtain one representative 60 msec pulse. This reduced 600 files, each with 600 values, to 30 files.

Prior to actually carrying out the averaging manipulation the individual pulses were plotted on the same set of axis to verify each has a similar waveshape. Each file was available for amplitude weighting and summation as discussed in Section 2. The array was formed by phase and/or amplitude tapering and summation of these averaged files. The quantitative assessment of the array would be determined by a measurable difference between the array signal with the plate waterborne and that with the plate removed.

5.2 Digitized Pulse Analysis

Since it is easier to visually integrate plots rather than tables of data Appendix C contains the library of digitized pulse shapes examined throughout the experiment. Each graph plots the 600 points for each pulse as a function of time. The digitized output from the analog/digital converter uses a 12 bit word giving 4096 possible values (2^{12}). The analog input varies from -5 volts to +5 volts resulting in an A/D conversion of 0.0024 volts per digitized unit value and values of digitized data from -2048 to +2048. The time scale is from 0 to .06 seconds or 60 msec. Figure 5.2 is a plot of the acoustic pulse at hydrophone 8 with the scattering plate in the water. As graphed the pulse is unaveraged and untapered.

The 0 time point corresponds to the pulse trigger and the actual acoustic signal will arrive at the array some time later based on the direct path distance from projector to hydrophone. A review of Section 3 will verify that this distance is approximately 23 feet to the hydrophone at array centerline, hydrophone #8. Using 4860 feet/second as the sound speed in water, the acoustic pulse along the direct path should arrive .0047 seconds

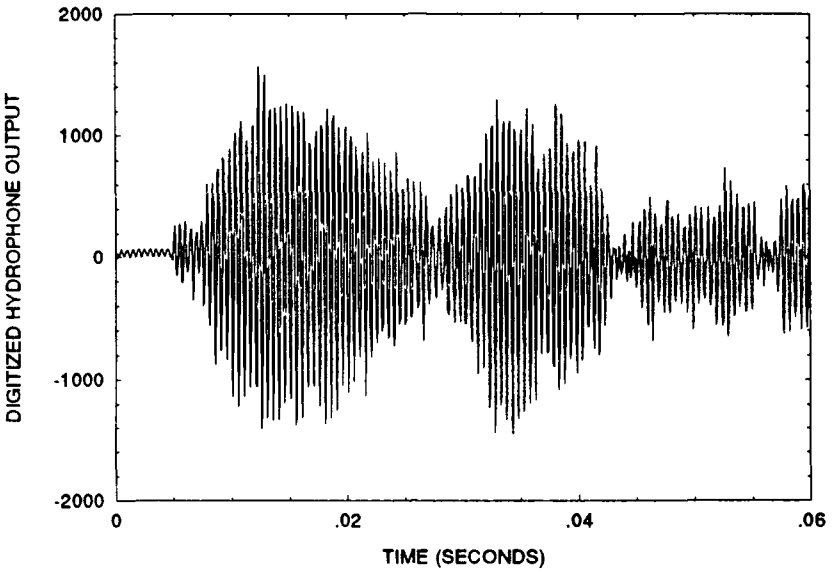


Figure 5.2 Hydrophone 8 digitized pulse

after trigger. Figure 5.2 supports this analysis.

Appendix C1 contains a series of pulse graphs corresponding to unaveraged, untapered hydrophone data. For each hydrophone position the two cases corresponding to the scattering plate are presented. A cursory review shows that extraction of one specific signal path from the pulse is very difficult. Any difference due to the plate is marginal at best. Additionally, this series of waveshapes vividly demonstrates the impact of the reverberant tank. The projected signal is 20 msec of 2 kHz sine waves. The received signals at each hydrophone are all significantly corrupted by the phase addition and subtraction resulting from multipath signals. The number of, and effects due to these multipath signals are the most significant problem associated with conducting array processing in the BBN reverberant tank.

A review of the 30 plots contained in Appendix C1 shows one potential problem. Hydrophone 11 is unique in that there is a significant difference between the case of scattering plate present and then removed. An initial evaluation might conclude that a multipath signal was responsible for almost perfect phase cancelation. It would be unlikely that the initial section of the direct path signal (4.7 - 5.7 msec) would also be canceled as it arrives at the hydrophone prior to any reflected path signal, an analysis tending to negate the proposed explanation. Additionally, it is expected that the free surface scattered signal should have a higher amplitude that the plate scattered signal. In order to examine the pulse, the scale of the axis were changed and the data replotted. Figure 5.3 shows hydrophone 11 with the scattering plate removed and the range of the digitized hydrophone output rescaled.

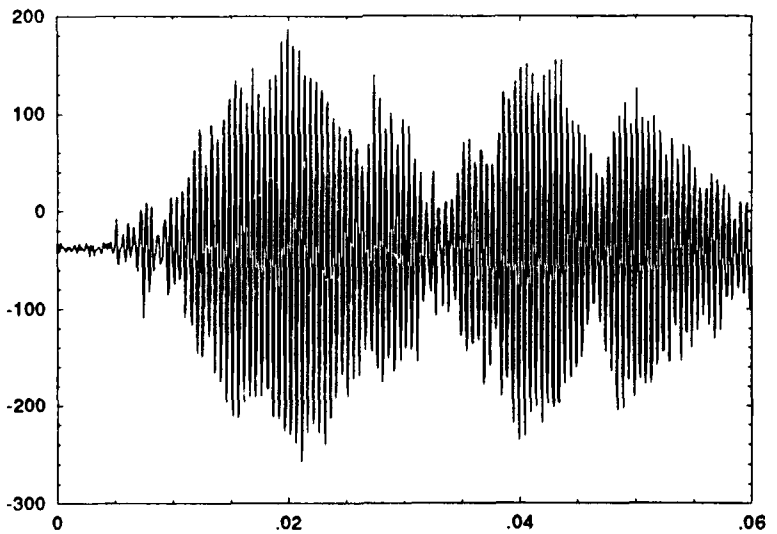


Figure 5.3 Hydrophone 11 digitized output

Comparing the pulse shape of Figure 5.3 to that of the same hydrophone with the scattering plate in the water contained in Appendix C1 shows that both have identical wave shapes. All twenty pulses used to average hydrophone 11 output with the scattering plate removed exhibit this phenomena. It is felt that during the data recording or translation phase of the experiment this signal was not amplified correctly. In order to minimize effect of the potentially corrupted data hydrophone 11 was averaged, tapered but not incorporated into the array beamforming. Since the major quantifiable assessment of the array is to examine the difference between array output with the scattering plate present and that with the plate removed, hydrophone 11 was not included in array processing.

Appendix C2 presents the same information in the same format but graphs the average of 20 pulses for each hydrophone. This series of plots when compared to Appendix C1 demonstrates the repeatability of each pulse. The pulse shape of the averaged signal is almost identical to that of each of the original pulses. As discussed in Section 3 the accuracy of a synthetic aperture is dependent on the repeatability of the active pulse. The averaging process is expected to minimize slight variations in the acoustic environment during the period of active interrogation and data recording.

Appendix C3 contains the averaged hydrophone data from Appendix C2 tapered using the Chebyshev element weights calculated by ARRAY and presented in Appendix B as well as Table 2.2. This set of amplitude taper weights should result in -30 dB side lobe levels with a 17.9° main lobe 3 dB beamwidth for a signal at 2 kHz. The associated beampattern for any signal at 5 Khz is also included in Appendix B for review.

5.3 Single Omnidirectional Hydrophone Analysis

Section i discussed the possibility of using a single hydrophone appropriately positioned to get a quantitative assessment of the effectiveness of an active control system. Because of the effects of the multipath signals the pulse length must be too short to give an adaptive control system sufficient time to respond. A 20 msec pulse while good for the control system results in a received signal like that in Figure 5.2.

Simple geometric calculation will show that the reflected signal from the plate arrives at approximately .0050 seconds (varying slightly depending on hydrophone position) after trigger. Figure 5.2 shows that it is virtually impossible to separate the desired signal from the received pulse even with apriori knowledge of the arrival time. The scattered signal arrives at the hydrophone while the direct path pulse is still passing through. Compounding the problem, the rear wall reflection also overlaps the time window of the desired signal. Time clipping simply will not work when long pulses are required and signals which are reflections of the active pulse can interfere.

5.4 Subtraction of the Reverberant Field

A second method to quantify the scattered signal involves subtracting the reverberant field. To accomplish this the received signal of an omnidirectional hydrophone taken without the plate in the water is subtracted from the signal at the same hydrophone position when the plate was present. The assumption is that the difference between the received pulses is due to the effect of the scattering plate.

Appendix D1 contains a series of pulse graphs based on this method. Each plot is the difference between the averaged received signal for a specific hydrophone position with the plate present and with the plate removed. As discussed above the scattered signal of interest is expected to arrive at the hydrophones in the vicinity of .0047 to .0053 seconds depending on hydrophone position.

A review of the plots contained in Appendix D1 indicates that this method does not result in an easily discernable pulse attributable to the scattered signal (5 to 25 msec along the time axis). The significant reduction in the value of the digitized hydrophone output indicates that the signal difference between the two cases is small. For long pulses in a reverberant facility the method of subtracting the reverberant field does not seem to provide a difference in received signals usable to identify the effects of an active control system.

5.5 Beamforming With Untapered, Averaged Signal

The beamforming process uses equations 2.9 and 2.15 to phase and amplitude taper each hydrophone signal and then sum the results. Since the array was physically positioned to direct the broadside beam to the target plate phase taper is not required. Using a rectangular amplitude weighting function of unity should result in the normalized array gain pattern of Figure 2.9 with the array characteristics in Table 2.2. This information is also contained in Appendix B. The array processing should result in coherent addition of the scattered signal and incoherent addition of noise and signals from the non-look directions. The expected first sidelobe should be reduced by -13 dB and the -3 dB beamwidth along the main response axis is 14°.

Figure 5.4 shows the resultant array output signal after processing with the scattering target plate in the water. There is reinforcement of the signal in the 5 msec to

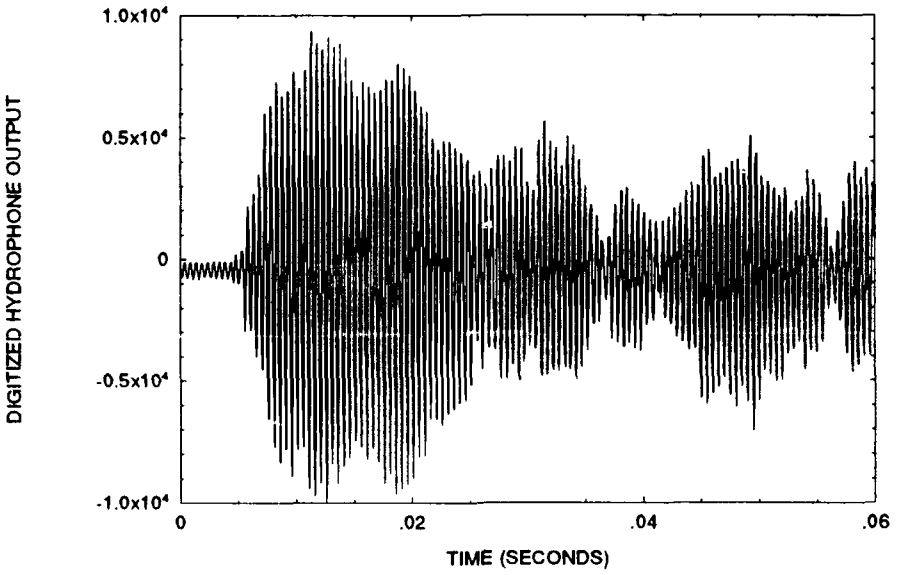


Figure 5.4 Untapered array output

30 msec range as was desired. Additionally as array theory indicates, there is an amplitude reduction relative to the desired signal outside of the time window established for the arrival of the scattered pulse. The initiation of the pulse on the graph occurs at 5.5 msec which is consistent with the geometry of the situation for a reflected path and later than the 4.7 msec pulse initiation seen in Figure 5.2 which is consistent with direct path arrival. There is some noticeable modulation of the pulse in the 5-25 msec window which is likely due to interference from alternate path signals. Considering the cone shape of the passive beam formed by a linear array this effect is expected.

Appendix D2 contains pulse plots of the processed array output wi. the plate both in and out of the tank. The pulse shapes are similar and show a slightly stronger signal when the scattering plate is absent. The results indicate that even simple non-amplitude tapered beamforming of a judiciously timeclipped signal will produce an output usable to

quantify the effectiveness of an active scattering control system on the target plate. The difference in magnitude of the pulse amplitude is difficult to see unless the pulses are plotted simultaneously.

5.6 Chebyshev Array

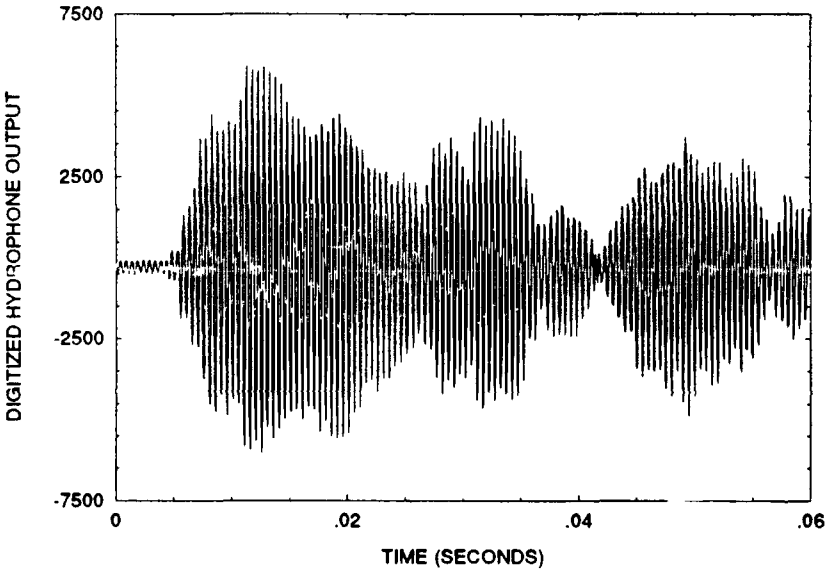


Figure 5.5 Chebyshev tapered array output

Array processing was repeated using the Chebyshev array amplitude weights of Table 2.2. These weighting factors should produce the beampattern shown in Appendix B. Figure 5.5 reproduces the plot of the tapered array output. The sidelobe levels all will be -30 dB with a -3 dB beamwidth of 17.9° on the main response axis. The results of this analysis are comparable to that of the untapered array and support the ability of array processing to allow the use of longer pulses in the active control of scattering

experiments.

Appendix D3 contains the plots of the processed array signal with the plate in and out of the reverberant facility for comparison. The pulse shapes and indicate phase addition at the desired time (5 to 25 msec) and phase cancellation of signals not from the desired look direction. The pulse begins to rise at 5.4 msec as expected for the reflected signal. The same modulation of the signal in the 5 to 25 msec window as was noted in the previous section is repeated. The amplitude of the signal when the plate is absent is noticeably larger than in the previous case. This is most likely due to the better sidelobe suppression expected in a Chebyshev array.

5.7 Power Spectral Density

In order to better visualize the effectiveness of an active scattering controller attached to the target plate it may be advantageous to examine the differences in power spectral density of the array output signal for several cases of interest. Such an analysis may result in a better interpretation of the effect the active controller has in a frequency band. As a validation of the array processing the power spectral density with the plate as a target can be compared with the same calculation with the plate absent. This would give an indication of the maximum change in the scattered signal that could occur should the active control system make the plate appear as a continuation of the free surface. The Lab Workbench environment can be used to calculate the power spectral density of the array output and to display the result graphically.

A single omnidirectional hydrophone, the untapered averaged array output and the Chebyshev array output were ported back to the MassComp computer by reversing the process discussed in Section 5.1. In the Lab Workbench environment the power spectral density was calculated, displayed and recorded. The virtual instrument created to accomplish this is schematically shown in Figure 5.6.

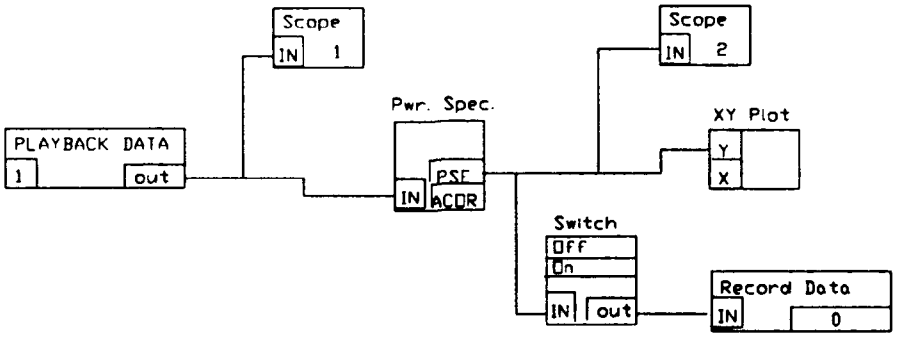


Figure 5.6 Power Spectral Density virtual instrument

The resulting plots of power spectral density are included in Appendix D4. The plots include both conditions of scattering plate position with a single hydrophone, the RECT window array and Chebyshev tapered array architectures. The difference in power spectral density for the two cases supports the contention that array beamforming will differentiate plate scattering from free surface scattering with pulse lengths as long as 20 msec. As expected the Chebyshev array is slightly superior to the rect weighted window. Table 5.1 summarizes the values extracted from the power spectral density graphs in Appendix D4.

Table 5.1 Power spectral density summary

Processing Method	Maximum Amplitude ($Volts^2/Hz$)	Frequency at Maximum Amplitude Hz
Omni hydrophone w/ plate	0.16296	2011
Omni hydrophone w/o plate	0.17670	2011
RECT window array w/ plate	2.25710	1982
RECT window array w/o plate	2.81620	1982
Chebyshev array w/ plate	1.12230	1982
Chebyshev array w/o plate	1.44990	1982

5.8 Summary

The ability to determine the change in the acoustic signal scattered off a plate located at the free surface is crucial to the study of the active control of scattering. Logistically and economically it is advantageous to conduct these experiments in a reverberant, acoustic tank. Due to the large number of multipath and reflected signals, the received pulse at one omnidirectional hydrophone can be significantly altered. A review of the received waveform at fifteen omnidirectional hydrophones documents the inability to accurately extract the desired signal scattered off the target plate. An attempt to simply subtract the reverberant field leaving only that part of the signal due to the plate was also ineffective.

The formation of a fifteen element array, properly phase tapered to position the main response axis on the target plate was successful in differentiating between the plate in the tank and the plate removed from the tank. The use of an interactive beamforming model, ARRAY, was integral in determining array characteristics and amplitude

weighting factors used in the post processing of the recorded data. The standard untapered array with -13 dB main sidelobe was successfully demonstrated as was a Chebyshev array with -30 dB sidelobes.

Creating the array aperture synthetically, by sequentially positioning the same hydrophone in fifteen different but known positions, was successfully demonstrated as a viable alternative to the costly method of building an array using fifteen separate hydrophones. Building a multi-element array would require testing to determine accurate phase and amplitude matching of the hydrophones prior to use.

6 Future Designs for Increased Capability

This report documents the successful use of a linear array to detect changes in scattered acoustic energy from a plate located at the free surface in a reverberant tank. As the desire to conduct acoustic testing in the reverberant tank increases so will the need to use more advanced means of array processing. If economically feasible, the next step should be to eliminate the synthetic aperture and build multi-element arrays which could be processed simultaneously. The MASSCOMP computer and TEAC data recorder have the capability to conduct this type of experiment once the required hydrophones and amplifiers are purchased.

In terms of array types the linear array should be replaced by the two dimensional plane array. This can significantly improve spatial filtering. The design tool ARRAY has the capability to be modified to allow plane array beamforming even to the extent of using different amplitude weighting functions in each dimension. The plotting routines are capable of contour and three dimensional plotting. Plane arrays will improve the signal to noise ratio allowing smaller scattering target fixtures.

The plate used as a scattering target in this study was a 4 ft X 4 ft steel plate. The plate anticipated for use in the active control of scattering experiment may be as small as 10 inches X 10 inches. This presents a significantly smaller target. The ease with which this study traded main response axis beamwidth for reduced sidelobe levels may no longer be a prudent consideration. The small effective cross section will require efforts to minimize beamwidth. Plane arrays and linear arrays of more elements may be required to obtain results comparable to this study. However, the physical dimensions of the BBN tank place an upper limit on how large the acoustic aperture can be.

High resolution beamforming methods should be investigated. Techniques such as Capon's [9], maximum likelihood method may provide improvements over classical beamforming techniques which justify their more sophisticated numerical techniques. The associated but separate techniques of adaptive signal processing discussed by Widrow [14] may be useful in the environment of the experimental tank facility, where the interfering signals are repetitive and essentially unchanging.

References

[1] Tucker, D.G. and Gazey, B.K., Applied Underwater Acoustics, Pergamon Press, Oxford, London, 1966.

[2] Urick, R. J., Principles Of Underwater Sound, McGraw Hill Book Company, New York, N.Y., 1975.

[3] Burdic, William S., Underwater Acoustic System Analysis,Prentice-Hall Inc., Englewood Cliffs, N.J., 1984.

[4] Clay, Clarance S. and Medwin, Herman, Acoustical Oceanography: Principles and Applications, John Wiley & Sons, New York, N.Y., 1977.

[5] Newland, D. E., An Introduction to Random Vibration and Spectral Analysis, Longman Inc., New York, N.Y., 1984.

[6] Collier, A. J., BMPAT: A Program for the Calculation and Display of the Response of Three Dimensional Arrays, Canadian Defense Research Establishment Atlantic Technical Communication 87/306, National Technical Information System (NTIS) AD-A181-739, 1987.

[7] Press, William H., Numerical Recipes in C, The Art of Scientific Computing, Cambridge University Press, Cambridge, 1988.

[8] Beyer, W.H. editor, Standard Mathematical Tables, 28th Edition, CRC Press Inc., Boca Raton, FL. 1987.

[9] Capon, J., *High Resolution Frequency-Wavenumber Spectrum Analysis*, Proceedings of the IEEE, 57, August 1969, 1408-1418.

[10] Hsu, K., Baggeroer, A., *Application of the Maximum-Likelihood Method (MLM) for Sonic Velocity Logging*, Geophysics, Vol 51, NO. 3, March 1986, 780-787.

[11] Kay, S., Marple, S., *Spectrum Analysis-A Modern Perspective*, Proceedings of the IEEE, Vol 69, NO. 11, November 1981, 1380-1419.

[12] Wernecke, S., D'Addario, L., *Maximum Entropy Image Reconstruction*, IEEE Transactions on Computers, Vol C-26, NO. 4, April 1977, 351-363.

[13] Pillai, S. U., Array Signal Processing, Springer-Verlag, New York, N.Y., 1989.

[14] Widrow, B., Sterns, S., Adaptive Signal Processing, Prentice-Hall, Englewood Cliffs, N.J., 1985.

[15] Elliott, D. F., The Handbook of Digital Signal Processing, Engineering Applications, Academic Press, Inc., San Diego, CA. 1987.

[16] Steinberg, B.D., Principles of Aperture and Array System Design, John Wiley & Sons, New York, N.Y., 1976.

[17] Dyer, I., Underwater Acoustics Class Notes, MIT Course 13.851 February, 1989

[18] Balanis, C.A., Antenna Theory Analysis and Design, Harper & Row, New York, N.Y., 1982.

[19] Pozar, D., Antenna Design Using Personal Computers, Artech House, Inc., Dedham, MA. 1985.

[20] Burns, M., Laxpati, S., Shelton, J., *A Comparative Study of Linear Array Synthesis Techniques Using a Personnel Computer*, IEEE Transactions Antennas Propagation, Vol. AP-32, August 1984, pp 884-887.

[21] Bresler, A., *A New Algorithm for Calculating the Current Distributions of Dolph-Chebyshev Arrays*, IEEE Transactions Antennas Propagation, Vol. AP-28, November 1980, pp 951-952.

[22] Abramowitz, M., Stegun, I., Handbook of Mathematical Functions, Dover Publications, New York, N.Y., 1968.

Appendix A Interactive Beamform Program ARRAY

```
/* ARRAY.C; MASTER CALLING PROGRAM TO CREATE AN ARRAY SYNTHESIS PROGRAM
USED TO CALCULATE AND GRAPHICALLY PRESENT PLOTS AND TABLES OF NORMALIZED
DIRECTIONAL GAIN PATTERNS FOR DISCRETE LINEAR ARRAYS OF OMNIDIRECTIONAL
ELEMENTS WITH USER SELECTED WINDOWING FUNCTIONS. */
/* rev 1: 2 APR 90 add TAYLOR WEIGHTING */
/* rev 2: 5 APR 90 add CHEBYSHEV WEIGHTING */
/* rev 3: 12 APR 90 add directivity and beamwidth */
/* rev 4: 17 APR 90 include switch for various video adapters */
/* rev 5: 18 APR 90 add output to file/printer support */

/* PREPROCESSOR DIRECTIVES */
#include <stdio.h>
#include <stdlib.h>
/* GLOBAL VARIABLE DEFINITIONS */
/* EXTERNAL VARIABLES DEFINED */
 int m1,m2;                    /* Number of elements in Array 1, 2 */
 float f1,f2;                  /* Signal frequency in Hz areray 1,2 */
 float d1,d2;                  /* Interelement spacing in feet array 1,2 */
 float theta1,theta2;          /* steer "look" angle broadside = 90 */
 float lambda1,lambda2;        /* signal wavelength */
 float c;                      /* sound speed in fps */
 int wint1,wint2;              /* window type */
 float *G1,*G2;                /* pointer for unit offset vector gain */
 float *win1,*win2;            /* pointer for vector of amplitude tapers */
 int i, j, k;                  /* loop counters */
 float di1,di2;                /* directivity for arrays 1, 2 */
 float bw1, bw2;               /* 3 dB main lobe beamwidth */
 float vmode,prn;              /* video mode and printer config flags */
 char inline[80];              /* used for config data entry */
 int ncnt;

/* FUNCTION DEFINITIONS */
  void menu();
/* MAIN PROGRAM LISTING */
void main()
{                              /* initiate main program */
/*************************************************************************
*     OPEN FILE TO OBTAIN CONFIGURATION DATA ON VIDEO SYSTEM AND        *
*     PRINTER HOOKUP                                                    *
*************************************************************************/
FILE *fp;
FILE *out;      /* output data file pointer */
fp = fopen("array.con","r");
if(fp==NULL)
  {
  system("cls");
  printf("*** ARRAY HAS NOT BEEN CONFIGURED FOR SYSTEM HARDWARE ***\n");
  printf("*** FILE ARRAY.CON NOT FOUND IN DEFAULT DIRECTORY ***\n");
  fp=fopen("array.con","a");
  mode:printf("\nSELECT AND ENTER GRAPHICS ADAPTER FROM MENU BELOW:\n");
  printf("\t1- CGA 4 Color\n\t2- EGA\n\t3- VGA\n\t4- HERCULES\n");
  printf("\nENTER GRAPHICS ADAPTER :");
  gets(inline);
  Stofa(inline,&vmode,&ncnt,1);
  switch((int) vmode)
    {
    case 1:
      vmode=1.0;
      break;
    case 2:
      vmode=6.0;
      break;
```

```
   case 3:
    vmode=8.0;
    break;
   case 4:
    vmode=12.0;
    break;
   default:
    printf(" INVALID CHOICE; PICK 1, 2, 3, OR 4.\n");
    goto mode;
   }
 system("cls");
 printf("\nSELECT AND ENTER PRINTER FROM MENU BELOW:\n");
 printf("\t1- EPSON MX\n\t2- EPSON LQ\n\t3- TOSHIBA 24 PIN\n\t4- HP LASER JET\n\t5- HP THINK
JET\n\t6- EPSON FX\n");
 gets(inline);
 stofa(inline,&prn,&ncnt,1);
 prn = prn-1.0;  /* shift indices to agree with InitSeGraphics() */
 fprintf(fp,"%f %f",vmode,prn);
 fclose(fp);
 }
if(fp!=NULL)
 {
 fscanf(fp,"%f %f",&vmode,&prn);
 }


/*************************************************************************
*    CONFIGURATION COMPLETE SHIFT TO PROGRAM EXECUTIVE MENU ( )        *
*************************************************************************/
menu();
}                       /* end main program     */
```

```
/* MENU1.C LISTING TO CREATE A MASTER EXECUTIVE ENVIRONMENT FOR ARRAY.C  */
/* PREPROCESSOR DIRECTIVES */
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include "worlddr.h"
#include "segraph.h"

#define PI 3.1415926
/* EXTERNAL VARIABLES DEFINED  */
 int m1,m2;                  /* Number of elements in Array  1, 2 */
 float f1,f2;                /* Signal frequency in Hz areray 1,2 */
 float d1,d2;                /* Interelement spacing in feet array 1,2 */
 float theta1,theta2;        /* steer "look" angle  broadside = 90 */
 float lambda1,lambda2;      /* signal wavelength          */
 float c;                    /* sound speed in fps         */
 int wint1,wint2;            /* window type                */
 float *G1, *G2;             /* pointer for unit offset vector gain */
 float *win1,*win2;          /* pointer for vector of amplitude tapers */
 float fm1, fm2, fwint1,fwint2;   /* floats of integer variables */
 int i, j, k;                /* loop counters              */
 int init;                   /* flag for initialization of plot windows */
 float di1,di2;
 float bw1,bw2;
 float vmode, prn;           /* video and printer configuration */
/* FUNCTION PROTOTYPES */
 void plot();
 void initial();
 void menu();
 float *vector();
 void nrerror();
 void free_vector();


 void menu()          /* MAIN EXECUTIVE FOR ARRAY SYNTHESIS */
  {
  char choice, chstr[2] ={0 ,0}, inline[81];
  int ncnt, mode,num;
  /* ALLOCATE MEMORY FOR UNIT OFFSET VECTORS */
 G1=vector(0,720);          /* DIRECTIONAL GAIN OVER 360 DEGREES */
 G2=vector(0,720);          /* DIRECTIONAL GAIN OVER 360 DEGREES */

 /* Initialize arrays and compute directional gain factors
   and plot to screen */
  initial();
 /* OPEN COMMAND WINDOW, INQUIRE IF ITERATION DESIRED */
 do                  /* master loop ends with quit program */
  {
  do
   {                  /* open main array iteration loop */
   SetCurrentWindow(1);
   ClearWindow();
   if(vmode!=12.)
    SetPlotBackground(2);
   BorderCurrentWindow(3);
   SelectColor(15);
   settextstyleXX(0.0,1);
   settextjustifyXX(0,0);
   TitleWindow("COMMAND WINDOW");
   setlinestyleXX(0,0,3);
```

```
  LabelGraphWindow(20,300,"ITERATE ARRAY <1/2>",0,0);
  choice = getch();
  chstr[0]=choice;
  LabelGraphWindow(850,300,chstr,0,0);
  CloseSEGraphics();
/*************************************************************************
*       DATA ENTRY SECTION TO ITERATE ARRAY PARAMETERS          *
*************************************************************************/
  system("cls");
  printf("\nENTER THE REQUIRED INFORMATION HIT RETURN TO ACCEPT <DEFAULT>\n\n");
  if(choice == '1')
    {
    printf("\tARRAY NUMBER ONE CHARACTERISTICS\n\n");
    printf("\t\t NUMBER OF ELEMENTS  <%d>: ",m1);
    gets(inline);
    Stofa(inline,&fm1,&ncnt,1);
    m1=(int) ceil(fm1);
    win1=vector(1,m1);
    printf("\t\t SIGNAL FREQUENCY (HZ) <%6.0f>: ",f1);
    gets(inline);
    Stofa(inline,&f1,&ncnt,1);
    printf("\t\t INTERELEMENT SPACING (FT) <%5.2f>: ",d1);
    gets(inline);
    Stofa(inline,&d1,&ncnt,1);
    printf("\t\t STEER DIRECTION (DEG) <%4.1f>: ",theta1);
    gets(inline);
    Stofa(inline,&theta1,&ncnt,1);
    printf("\t\t INDICATE DESIRED AMPLITUDE TAPER FUNCTION \n");
    printf("\t***CURRENT PROGRAM REQUIRES SYMMETRY ABOUT ARRAY CENTER***\n");
    printf("\t\t\t 1- RECTANGULAR\n");
    printf("\t\t\t 2- USER DEFINED\n");
    printf("\t\t\t 3- RAISED COSINE\n");
    printf("\t\t\t 4- RAISED COSINE SQUARED\n");
    printf("\t\t\t 5- TAYLOR\n");
    printf("\t\t\t 6- CHEBYSHEV\n");
    printf("\t\t ENTER MENU NUMBER FOR TAPER FUNCTION <%d>  ",wint1);
    gets(inline);
    Stofa(inline,&fwint1,&ncnt,1);
    wint1=(int) fwint1;
    printf("\t\t ENTER SOUND SPEED (c) <%5.1f>: ",c);
    gets(inline);
    Stofa(inline,&c,&ncnt,1);
    lambda1 = c/f1;
    }
  if(choice == '2')
    {
    printf("\tARRAY NUMBER TWO CHARACTERISTICS\n\n");
    printf("\t\t NUMBER OF ELEMENTS  <%d>: ",m2);
    gets(inline);
    Stofa(inline,&fm2,&ncnt,1);
    m2=(int) ceil(fm2);
    win2=vector(1,m2);
    printf("\t\t SIGNAL FREQUENCY (HZ) <%6.0f>: ",f2);
    gets(inline);
    Stofa(inline,&f2,&ncnt,1);
    printf("\t\t INTERELEMENT SPACING (FT) <%5.2f>: ",d2);
    gets(inline);
    Stofa(inline,&d2,&ncnt,1);
    printf("\t\t STEER DIRECTION (DEG) <%4.1f>: ",theta2);
    gets(inline);
    Stofa(inline,&theta2,&ncnt,1);
    printf("\t\t INDICATE DESIRED AMPLITUDE TAPER FUNCTION \n");
```

```
    printf("\t***CURRENT PROGRAM REQUIRES SYMMETRY ABOUT ARRAY CENTER***\n");
    printf("\t\t\t 1- RECTANGULAR\n");
    printf("\t\t\t 2- USER DEFINED\n");
    printf("\t\t\t 3- RAISED COSINE\n");
    printf("\t\t\t 4- RAISED COSINE SQUARED\n");
    printf("\t\t\t 5- TAYLOR\n");
    printf("\t\t\t 6- CHEBYSHEV\n");
    printf("\t\t ENTER MENU NUMBER FOR TAPER FUNCTION <%d> ",wint2);
    gets(inline);
    Stofa(inline,&fwint2,&ncnt,1);
    wint2=(int) fwint2;
    printf("\t\t ENTER SOUND SPEED (c) <%5.1f>: ",c);
    gets(inline);
    Stofa(inline,&c,&ncnt,1);
    lambda2 = c/f2;
    }
/*****************************************************************************
*      INITIATE GRAPHICS MODE AND PLOT REVISED ARRAY BEAM PATTERN      *
*****************************************************************************/
 if(choice == '1')
   beam(m1,d1,lambda1,theta1,wint1,G1,win1,&di1,&bw1);
 else
   beam(m2,d2,lambda2,theta2,wint2,G2,win2,&di2,&bw2);
 InitSEGraphics((int) vmode);
 plot(2,1,wint1,G1);
 plot(2,2,wint2,G2);
 free_vector(win1,0,m1);
 free_vector(win2,0,m2);


 SetCurrentWindow(1);
 ClearWindow();
 if(vmode!=12.)
   SetPlotBackground(2);
 BorderCurrentWindow(3);
 SelectColor(15);
 settextstyleXX(0,0,1);
 settextjustifyXX(0,0);
 TitleWindow("COMMAND WINDOW");
 setlinestyleXX(0,0,3);
 LabelGraphWindow(20,600,"ITERATE ? <y/n>",0,0);
 choice = getch();
 chstr[0]=choice;
 LabelGraphWindow(850,600,chstr,0,0);
 } while (choice != 'n');      /* close array iteration loop */
 ClearWindow();
 LabelGraphWindow(20,600,"OUTPUT DATA ? <y/n>",0,0);
 choice = getch();
 chstr[0]=choice;
 LabelGraphWindow(850,600,chstr,0,0);
 if(choice=='y'|| choice=='Y')
   {
   ClearWindow();
   LabelGraphWindow(20,600,"Array 1 2 both? <1/2/b>",0,0);
   choice = getch();
   chstr[0]=choice;
   LabelGraphWindow(850,400,chstr,0,0);
   switch(choice)
      {
      case '1':
        output(f1,c,m1,d1,lambda1,theta1,wint1,di1,bw1,win1,G1);
        break;
```

```
        case '2':
          output(f2,c,m2,d2,lambda2,theta2,wint2,di2,bw2,win2,G2);
          break;
        case 'b':
          output(f1,c,m1,d1,lambda1,theta1,wint1,di1,bw1,win1,G1);
          output(f2,c,m2,d2,lambda2,theta2,wint2,di2,bw2,win2,G2);
          break;
        }
     plot(2,1,wint1,G1);
     plot(2,2,wint2,G2);
     }
    SetCurrentWindow(1);
    ClearWindow();
    if(vmode!=12.)
     SetPlotBackground(2);
    BorderCurrentWindow(3);
    SelectColor(15);
    settextstyleXX(0,0,1);
    settextjustifyXX(0,0);
    TitleWindow("COMMAND WINDOW");
    setlinestyleXX(0,0,3);
    LabelGraphWindow(20,600,"QUIT ? <y/n>",0,0);
    choice = getch();
    chstr[0]=choice;
    LabelGraphWindow(850,600,chstr,0,0);
    } while (choice != 'y');       /* close array master loop */

   CloseSEGraphics();
   free_vector(G1,0,720);
   free_vector(G2,0,720);
   }              /* end menu() the main executive program */


/*************************************************************************
*    CALCULATE ARRAY PARAMETERS AND PLOT DIRECTIONAL GAIN USING      *
*    INITIALIZATION VALUES FOR ARRAYS 1 AND 2, (see ARRAY.C)         *
*************************************************************************/
void initial()
 {
 int mode;          /* set screen mode to single or dual plot */
 int num;           /* set array to be plotted   */
/* Initialize both arrays: 2KHZ, 10/20 elements, steer 90/45 degrees, 1/2
   wavelength interelement distance, celerity 4860 fps, with rect window
   function of amplitude 1. */
m1=10;
fm1=10.;
m2=20;
fm2=20.;
c=4860.;
f1=2000.;
f2=2000.;
lambda1=c/f1;
lambda2=c/f2;
d1=lambda1/2.;
d2=lambda2/2.;
theta1=90.;
theta2=45.;
wint1=1;
wint2=1;
fwint1=1.;
fwint2=1.;
```

```
mode=2;                    /* set screen to dual plot mode */
if(vmode ==12.000)
  system("msherc");
printf("Initiating Default Graphics Window");
InitSEGraphics((int) vmode);
beam(m1,d1,lambda1,theta1,1,G1,win1,&di1,&bw1);
plot(mode,1,wint1,G1);
beam(m2,d2,lambda2,theta2,1,G2,win2,&di2,&bw2);
plot(mode,2,wint2,G2);

}
```

```
/*************************************************************************
*    BEAM.C CALCULATES THE NORMALIZED DIRECTIONAL GAIN PATTERN          *
*    G[THETA] FOR ANGLES 0<theta<179 CONSIDERING AMPLITUDE TAPER        *
*    THEN COMPLETES VALUES FOR ANGULAR SPACE BASED ON SYMMETRY.         *
*************************************************************************/
/* PREPROCESSOR DIRECTIVES */
#include <stdio.h>
#include <setjmp.h>
#include <math.h>
#include <stdlib.h>
#define PI 3.1415926
/* EXTERNAL VARIABLES DEFINED */
 int m;                  /* Number of elements in Array  1, 2 */
 float f;                /* Signal frequency in Hz areray 1,2 */
 float d;                /* Interelement spacing in feet array 1,2 */
 float theta;            /* steer "look" angle  broadside = 90 */
 float lambda;           /* signal wavelength          */
 float c;                /* sound speed in fps         */
 int wint;               /* window type                */
 float *G;               /* pointer for unit offset vector gain */
 float *win;             /* pointer for vector of amplitude tapers */
 int i, j, k;            /* loop counters              */
 int wint;               /* amplitude taper type marker */
 char inline[81];        /* data entry */
 char ncnt;
 float *di;
 float *bw;
/* FUNCTION PROTOTYPES */
 void beam();
 float bessi0();
 float *vector();
 double *dvector();
 void nrerror();
 void free_vector();
 float dir();
 float dbdown();
void beam(m,d,lambda,theta,wint,G,win,di,bw)
 int m,wint;
 float d,lambda,theta,G[],win[],*di,*bw;
 {                       /* open beam() function */
 char ch;
 float num, den, ftheta[721], norm, fact, weight;
 float height, proj, x;  /* window height and aperature projection */
 double cheb,cheb1, side, angle:    /* chebyshev polynomial, sidelobe  */
 int flag, thetai:
 double *a;  /* chebyshev temporary weights */
 thetai= (int) ceil(theta);
 theta = theta*PI/180.;
 if(wint==1)    /* loop for rect window uses [sin mx/(m sinx)] for speed */
  {
   norm=0;
   for (i=0;i<=360;i++)  /* for spacial angles 0<theta<179 */
     {
     angle= i*PI/360.;
     den =PI*d/lambda*(cos(angle)-cos(theta));
     ftheta[i]=1.;
     if(fabs(den)>.00001)
       {
       ftheta[i]=sin(m*den)/(m*sin(den));
       }
     ftheta[i]=fabs(ftheta[i]);
     if(ftheta[i]>norm)
```

```
    {
    norm=ftheta[i];
    }
  }
  for(i=361;i<=720;i++)
    {
    ftheta[i]=ftheta[720-i];
    }
  for(i=0;i<=720;i++)
    {
    ftheta[i] = ftheta[i]/norm;
    ftheta[i]=ftheta[i]*ftheta[i];
    if(ftheta[i]<.0000001)
       {
       ftheta[i] = .0000001;
       }
    G[i]=10*log10(ftheta[i]);
       }
 /* Determine 3 db main beamwidth & DI */
   *di=dir(ftheta);
   *bw=dbdown(G,theta);

  } /* close  conditional for rect window function */
if(wint == 2) /* user defined amplitude weighting window  */
  {
  system("cls");
  printf("\tARRAY AMPLITUDE WEIGHTS\n\n");
  printf("\t\t NUMBER OF ELEMENTS  <%d>:\n ",m);
  repeat: for(j=1;j<=m;j++)
    {
    printf("\n\t ELEMENT %d  WEIGHT : ",j);
    gets(inline);
    Stofa(inline,&win[j],&ncnt,1);
    }
  printf("\n\n\tVERIFY ENTERED WEIGHT VALUES\n\n");
  for(j=1;j<=m;j++)
    {
    printf("\t ELEMENT %d  WEIGHT : %8.4f\n ",j,win[j]);
    }
  printf("ARE THESE VALUES CORRECT <y/n>?  ");
  if(ch = getche() == 'n')
    {
    goto repeat;
    }
  }   /* close user defined amplitude window beam calculations  */
if(wint == 3 || wint == 4)  /* RAISED COSINE AMPLITUDE WEIGHT CALCULATION */
  {
  system("cls");
  if(wint == 3)
    printf("\tRAISED COSINE ARRAY AMPLITUDE WEIGHTS\n\n");
  else
    printf("\tRAISED COSINE SQUARED ARRAY AMPLITUDE WEIGHTS\n\n");
  printf("\t\t NUMBER OF ELEMENTS  <%d>:\n ",m);
  repeat3: printf("ENTER THE FRACTIONAL PEDESTAL HEIGHT (0.0 - 1.0)\n");
  printf("NOTE A PEDESTAL HEIGHT OF 1.0 RETURNS A RECT WINDOW\n");
  printf("\t HEIGHT: ",height);
  gets(inline);
  Stofa(inline,&height,&ncnt,1);
  for(j=1;j<=m;j++)
    {
    proj = 1.-(j-1)*2./(m-1.);
    if(wint == 3)   /* raised cos weight calculation */
```

```
      win[j] = height + (.5-height/2.)*cos(proj*PI);
   else          /* raised cos squared weights */
      win[j] = height + (1.-height)*cos(proj*PI/2.)*cos(proj*PI/2.);
   }
  if(2* (m/2) != m)
   {
   num = (m-1.)/2.+1.;  /* odd number of elements */
   win[(int) num] = win[(int) num]/2.0;
   }
  printf("\n\n\tVERIFY CALCULATED WEIGHT VALUES\n\n");
  for(j=1;j<=m;j++)
   {
   printf("\t ELEMENT %d  WEIGHT : %8.4f\n ",j,win[j]);
   }
  printf("ARE THESE VALUES CORRECT <y/n>?  ");
  if(ch = getche() == 'n')
   {
   goto repeat3;  /* reenter pedestal fractional height */
   }
 }  /* close cosine amplitude weight calculations */
if(wint == 5)  /* Taylor weight factor calculations */
 {
 system("cls");
 printf("\tTAYLOR FUNCTION ARRAY AMPLITUDE WEIGHTS\n\n");
 printf("\t\t NUMBER OF ELEMENTS  <%d>:\n ",m);
 repeat5: printf("ENTER THE TAYLOR PARAMETER (0 - 6):\n");
 printf("NOTE LARGER VALUES GIVE SMALL SIDELOBES AND WIDE MAIN LOBE\n");
 printf("\t TAYLOR PARAMETER: ",height);  /* REUSE HEIGHT variable*/
 gets(inline);
 Stofa(inline,&height,&ncnt,1);
 for(j=1;j<=m;j++)
  {
  proj = 1.-(j-1)*2./(m-1.);
  num = height*sqrt(1-proj*proj);
  win[j]=bessi0(num);   /* weight = mod bessel order 0 */
  }
 if(2* (m/2) != m)
  {
  num = (m-1.)/2.+1.;  /* odd number of elements */
  win[(int) num] = win[(int) num]/2.0;
  }
 printf("\n\n\tVERIFY CALCULATED WEIGHT VALUES\n\n");
 for(j=1;j<=m;j++)
  {
  printf("\t ELEMENT %d  WEIGHT : %8.4f\n ",j,win[j]);
  }
 printf("ARE THESE VALUES CORRECT <y/n>?  ");
 if(ch = getche() == 'n')
  {
  goto repeat5;  /* reenter Taylor Parameter */
  }
 }  /* close Taylor function weight calculation loop */
if(wint == 6)  /* Dolph-Chebyshev weight calculations */
 {
 system("cls");
 printf("\tDOLPH-CHEBYSHEV FUNCTION ARRAY AMPLITUDE WEIGHTS\n\n");
 printf("\t\t NUMBER OF ELEMENTS  <%d>:\n ",m);
 repeat6: printf("ENTER THE DESIRED SIDELOBE LEVEL ");
 gets(inline);
 Stofa(inline,&height,&ncnt,1); /* reuse variable again */
 /* even or odd # elements  */
```

```
flag = 0;
if(2* (m/2) == m)
 flag = 1;  /* even number of elements */
if(flag == 1)
 num =m/2.;
else
 num = (m-1.)/2.;
k=ceil(num);
a=dvector(0,m);
/* Compute chebyshev polynomial */
side=pow(10.,(height/20.));
cheb1=((log(side+sqrt(side*side-1)))/(m-1));
cheb=.5*(exp(cheb1)+exp(-cheb1));
angle=1.-1./(cheb*cheb);
for(i=1;i<=k-flag;i++)
 {
 fact=1.;
 for(j=1;j<=i-1;j++)
    {
    den = (float)j*(m-1-2*i+j)/((i-j)*(i+1-j));
    fact=fact*angle*den+1;
    }
 a[k-i]=(m-1)*angle*fact;
 }
a[k]=1.;
norm=a[flag];
if(flag == 0)
 win[k+1]=1;
for(j=1;j<=k;j++)
 {
 win[j]=a[k+1-j]/norm;
 win[m+1-j]=win[j];
 }
if(2* (m/2) != m)
 {
 num = (m-1.)/2.+1.;  /* odd number of elements */
 win[(int) num] = win[(int) num]/2.0;
 }
printf("\n\n\tVERIFY CALCULATED WEIGHT VALUES\n\n");
for(j=1;j<=m;j++)
 {
 printf("\t ELEMENT %d  WEIGHT : %8.4f\n ",j,win[j]);
 }
printf("ARE THESE VALUES CORRECT <y/n>?  ");
if(ch = getche() == 'n')
 {
 goto repeat6;  /* reenter sidelobe levels */
 }
free_dvector(a,0,num);
}   /* close Chebyshev weight loop  */


if(wint > 1)  /* control loop to calculate beam non rect window */
{
for(i=0;i<=720;i++) /* rezero ftheta vector */
 {
 ftheta[i]=0.0;
 }
/* even or odd # elements  */
```

```
 flag = 0;
 if(2* (m/2) == m)
  {
  flag = 1;  /* even number of elements */
  }
 if(flag == 1)
  {
  num =m/2.;
  }
 else
  {
  num = (m-1.)/2.+1.;
  }
 norm = 0.;
 for(i=0;i<=360;i++)
  {
  for(j=1;j<=(int)num;j++)
   {
   if(flag==0)
       fact = (j-1.);
   if(flag==1)
        fact = (j-.5);
   weight = win[(int) num+1-j];
   ftheta[i]=ftheta[i]+weight*cos(fact*2.*PI*d/lambda*(cos(i*PI/360.)-cos(theta)));
   }
  ftheta[i]=fabs(ftheta[i]);
  if(ftheta[i] > norm)
   norm = ftheta[i];
  }
 for(i=361;i<=720;i++)
  ftheta[i] = ftheta[720-i];
 for(i=0;i<=720;i++)
  {
  ftheta[i]=ftheta[i]/norm;
  ftheta[i]=ftheta[i]*ftheta[i];
  if(ftheta[i]<= .0000001)
   ftheta[i]=.0000001;
  G[i]=10*log10(ftheta[i]);
  }
  *di=dir(ftheta);
  *bw=dbdown(G,thetai);
 }  /*  close beam calc control loop */
}   /*  close beam function */
/*****************************************************************************
*  FUNCTION BESSIO(X) CALCULATES MODIFIED BESSEL FUNCTIONS OF ZERO ORDER  *
*  ADAPTED FROM NUMERICAL RECIPES IN C BY PRESS ET. AL.              *
*****************************************************************************/
float bessi0(x)
float x;
{
float ax, ans;
double y;
if((ax = fabs(x)) < 3.75)
 {
 y=x/3.75;
 y*=y;
 ans=1.0+y*(3.5156229+y*(3.0899424+y*(1.2067492+y*(0.2659732+y*
 (0.360768e-1+y*0.45813e-2)))));
 }
else
 {
 y=3.75/ax;
```

```
 ans=(exp(ax)/sqrt(ax))*(0.39894228+y*(0.1328592e-1+y*(0.225319e-2
 +y*(-0.157565e-2+y*(0.916281e-2+y*(-0.2057706e-1+y*(0.2635537e-1
 +y*(-0.1647633e-1+y*0.392377e-2)))))));
  }
return ans;
}        /* close modified bessel function bessi0() */
/*************************************************************************
*    CALCULATION OF DIRECTIVITY BY NUMERICAL INTEGRATION OF ARRAY      *
*    FACTOR USING EXTENDED SIMPSON'S RULE                              *
*************************************************************************/
float dir(ftheta)
  float ftheta[];
  {
  float s;
  int i;
  s=0.0;
  for(i=0;i<=718;i=i+2)
   {
   s=s+1.0/(3.0)*.5/360.*(ftheta[i]+4.0*ftheta[i+1]+ftheta[i+2]);
   }
  s=10*log10(1./s);
  return s;
  }


/*************************************************************************
*    FUNCTION TO CALCULATE 3 dB MAIN LOBE BEAMWIDTH                     *
*************************************************************************/
float dbdown(G,thetai)
 float G[];
 int thetai;
 {
 float s,sb;
 s=0.0;
 for(i=1;i<=90;i++)
  {
  sb=s;
  s=G[2*thetai]-G[(2*thetai)+i];
  if(fabs(s)>=3.)
   return (i-1)+(3-sb)/(s-sb);
  }
 printf(" 3 dB down main lobe width exceeds 90 degrees\n");
 printf("Press any key to continue");
 getch();
 }
```

```
*************************************************************************
*   PLOT.C A FUNCTION TO PLOT THE CURVES OF NORMALIZED DIRECTIONAL BEAM  *
*   PATTERN CALCULATED IN BEAM.C                                        *
*************************************************************************/
/* PREPROCESSOR DIRECTIVES */
#include <stdio.h>
#include<graph.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include "worlddr.h"
#include "segraph.h"
/* EXTERNAL VARIABLES DEFINED */
 float *G;   /* pointer to vector of array gain values */
 int mode;   /* integer to select screen mode single vs. dual graph */
 int num;  /* select which array to plot */
 int m1,m2;                  /* Number of elements in Array 1, 2 */
 float f1,f2;                /* Signal frequency in Hz areray 1,2 */
 float d1,d2;                /* Interelement spacing in feet array 1,2 */
 float theta1,theta2;        /* steer "look" angle broadside = 90 */
 float lambda1,lambda2;      /* signal wavelength */
 float c;                    /* sound speed in fps */
 int wint1,wint2;            /* window type */
 float *G1, *G2;             /* pointer for unit offset vector gain */
 float *win1,*win2;          /* pointer for vector of amplitude tapers */
 int i, j, k;                /* loop counters */
 int init;                   /* flag for initialization of plot windows */
 float di1,di2;
 float bw1,bw2;
 float vmode,prn;
/* FUNCTION PROTOTYPES */
 void plot();
 void plot(mode, num , wint, G)
  float G[];
  int mode, num, wint;
  {
  int i, maxX, maxY,plt,data;
  float angle[721];     /* vector of x axis data, degrees */
  char tempstr[20];
  char xstr[20], choice;
  for(i=0;i<=720;i++)
   {
   angle[i]=i/2.;
   }
   GetMaxCoords(&maxX,&maxY);
   DefGraphWindow(maxX/3,1,maxX,maxY/2.8);
   DefGraphWindow(maxX/3,maxY/2,maxX,maxY,10);
   DefGraphWindow(0,0,Round(maxX/3.0),maxY/2.,7);
   DefGraphWindow(0,Round(maxY/2.),Round(maxX/3.),maxY/1.2,9);
   DefGraphWindow(0,Round(maxY/1.2),Round(maxX/3.),maxY,1);
   SetWin2PlotRatio(8,.2,0.14,0.05,0.14);
   SetWin2PlotRatio(10,.2,0.14,0.05,0.14);
   SetWin2PlotRatio(7,0,0,0,0);
   SetWin2PlotRatio(9,0,0,0,0);
   SetWin2PlotRatio(1,0,0,0,0);


  if(num == 1)
   {
   SetCurrentWindow(8);
   ClearWindow();
   SetAxesType(0,0);
```

```
if(vmode!=12.)
  SetPlotBackground(3);
SelectColor(15);
AutoAxes(angle,G,720,1);
TitleWindow("ARRAY #1 BEAM PATTERN");
TitleXAxis("Theta (degrees)");
TitleYAxis("G(Theta) (db)");
BorderCurrentWindow(4);
SetCurrentWindow(7);
ClearWindow();
if(vmode!=12.)
  SetPlotBackground(3);
BorderCurrentWindow(4);
SelectColor(15);
setlinestyleXX(0,0,1);
TitleWindow("ARRAY #1");
settextjustifyXX(0,0);
sprintf(xstr,"%6.0f",f1);
strcpy(tempstr,"Freq (Hz):      ");
strcat(tempstr, xstr );
LabelGraphWindow(20,880,tempstr,0,0);
sprintf(xstr."%6.0f",c);
strcpy(tempstr,"Celerity (fps): ");
strcat(tempstr, xstr );
LabelGraphWindow(20,830,tempstr,0,0);
sprintf(xstr,"%3d",m1);
strcpy(tempstr,"# elements: ");
strcat(tempstr, xstr );
LabelGraphWindow(20,780,tempstr,0,0);
sprintf(xstr,"%5.2f",d1);
strcpy(tempstr,"spacing (ft): ");
strcat(tempstr, xstr );
LabelGraphWindow(20,730,tempstr,0,0);
sprintf(xstr,"%4.2f",lambda1);
strcpy(tempstr,"Wavelength (ft): ");
strcat(tempstr, xstr );
LabelGraphWindow(20,680,tempstr,0,0);
sprintf(xstr,"%4.1f",theta1);
strcpy(tempstr,"Look Angle (deg): ");
strcat(tempstr, xstr );
LabelGraphWindow(20,630,tempstr,0,0);
switch(wint)
 {
 case 1:
    strcpy(tempstr,"Window: RECTANGULAR");
    LabelGraphWindow(20,580,tempstr,0,0);
    break;
 case 2:
    strcpy(tempstr,"Window: USER");
    LabelGraphWindow(20,580,tempstr,0,0);
    break;
 case 3:
    strcpy(tempstr,"Window: COSINE");
    LabelGraphWindow(20,580,tempstr,0,0);
    break;
 case 4:
    strcpy(tempstr,"Window: COSINE SQUARED");
    LabelGraphWindow(20,580,tempstr,0,0);
    break;
 case 5:
    strcpy(tempstr,"Window: TAYLOR");
    LabelGraphWindow(20,580,tempstr,0,0);
```

```
        break;
      case 6:
        strcpy(tempstr,"Window:  CHEBYSHEV");
        LabelGraphWindow(20,580,tempstr,0,0);
        break;
      }
    sprintf(xstr,"%5.3f",di1);
    strcpy(tempstr,"Directivity(dB): ");
    strcat(tempstr, xstr );
    LabelGraphWindow(20,530,tempstr,0,0);
    sprintf(xstr,"%2.1f",bw1);
    strcpy(tempstr,"3 dB beam width: ");
    strcat(tempstr, xstr );
    LabelGraphWindow(20,480,tempstr,0,0);

    SetCurrentWindow(8);
    ClearGraph();
    if(vmode!=12.)
      SetPlotBackground(3);
    SelectColor(4);
    DrawGrid(6);
    LinePlotData(angle,G,721,15,0);  /* command to plot array #1 G(theta) */
    }  /* close num = 1 */
  else
    {
    SetCurrentWindow(10);
    ClearWindow();
    SetAxesType(0,0);
    if(vmode!=12.)
      SetPlotBackground(14);
    SelectColor(15);
    AutoAxes(angle,G,720,1);
    TitleWindow("ARRAY #2 BEAM PATTERN");
    TitleXAxis("Theta (degrees)");
    TitleYAxis("G(Theta) (db)");
    BorderCurrentWindow(3);
    SetCurrentWindow(9);
    ClearWindow();
    if(vmode!=12.)
      SetPlotBackground(14);
    BorderCurrentWindow(3);
    SelectColor(1);
    setlinestyleXX(0,0,1);
    TitleWindow("ARRAY #2");
    settextjustifyXX(0,0);
    sprintf(xstr,"%6.0f",f2);
    strcpy(tempstr,"Freq (Hz):      ");
    strcat(tempstr, xstr );
    LabelGraphWindow(20,850,tempstr,0,0);
    sprintf(xstr,"%6.0f",c);
    strcpy(tempstr,"Celerity (fps): ");
    strcat(tempstr, xstr );
    LabelGraphWindow(20,780,tempstr,0,0);
    sprintf(xstr,"%3d",m2);
    strcpy(tempstr,"# elements: ");
    strcat(tempstr, xstr );
    LabelGraphWindow(20,710,tempstr,0,0);
    sprintf(xstr,"%5.2f",d2);
    strcpy(tempstr,"spacing (ft): ");
    strcat(tempstr, xstr );
    LabelGraphWindow(20,640,tempstr,0,0);
    sprintf(xstr,"%4.2f",lambda2);
```

```
strcpy(tempstr,"Wavelength (ft): ");
strcat(tempstr, xstr );
LabelGraphWindow(20,570,tempstr,0,0);
sprintf(xstr,"%4.1f",theta2);
strcpy(tempstr,"Look Angle (deg): ");
strcat(tempstr, xstr );
LabelGraphWindow(20,490,tempstr,0,0);
switch(wint)
 {
 case 1:
    strcpy(tempstr,"Window: RECTANGULAR");
    LabelGraphWindow(20,410,tempstr,0,0);
    break;
 case 2:
    strcpy(tempstr,"Window: USER");
    LabelGraphWindow(20,410,tempstr,0,0);
    break;
 case 3:
    strcpy(tempstr,"Window: COSINE");
    LabelGraphWindow(20,410,tempstr,0,0);
    break;
 case 4:
    strcpy(tempstr,"Window: COSINE SQUARED");
    LabelGraphWindow(20,410,tempstr,0,0);
    break;
 case 5:
    strcpy(tempstr,"Window: TAYLOR");
    LabelGraphWindow(20,410,tempstr,0,0);
    break;
 case 6:
    strcpy(tempstr,"Window: CHEBYSHEV");
    LabelGraphWindow(20,410,tempstr,0,0);
    break;
 }
sprintf(xstr,"%5.3f",di2);
strcpy(tempstr,"Directivity(dB): ");
strcat(tempstr, xstr );
LabelGraphWindow(20,330,tempstr,0,0);
sprintf(xstr,"%2.1f",bw2);
strcpy(tempstr,"3 dB beam width: ");
strcat(tempstr, xstr );
LabelGraphWindow(20,250,tempstr,0,0);
SetCurrentWindow(10);
ClearGraph();
if(vmode!=12.)
 SetPlotBackground(14);
SelectColor(2);
DrawGrid(6);
LinePlotData(angle,G,720,1,0); /* COMMAND TO PLOT ARRAY 2 DATA */
 } /* else */
} /* close plot function */
```

```
/*************************************************************************
*    OUTPUT.C CONTROLS ARRAY DATA OUTPUT TO SELECTED FILE
*    OR TO PRINTER
*************************************************************************/
/* PREPROCESSOR DIRECTIVES */
/* PREPROCESSOR DIRECTIVES */
#include <stdio.h>
#include <setjmp.h>
#include <math.h>
#include <stdlib.h>
#include <graph.h>
#include <string.h>
#include "worlddr.h"
#include "segraph.h"


/* EXTERNAL VARIABLES DEFINED  */
 int m;                  /* Number of elements in Array  1, 2 */
 float f;                /* Signal frequency in Hz areray 1,2 */
 float d;                /* Interelement spacing in feet array 1,2 */
 float theta;            /* steer "look" angle  broadside = 90 */
 float lambda;           /* signal wavelength          */
 float c;                   /* sound speed in fps        */
 int wint;               /* window type               */
 float *G;               /* pointer for unit offset vector gain */
 float *win;             /* pointer for vector of amplitude tapers */
 int i, j, k;               /* loop counters             */
 int wint;                  /* amplitude taper type marker */
 char inline[81];        /* data entry */
 char ncnt;
 float di;
 float bw;
 void output();
 float prn;
void output(f,c,m,d,lambda,theta,wint,di,bw,win,G)
 float f,c,d,lambda,theta,di,bw;
 float G[],win[];
 int m,wint;
 {
 int error;
 int i,maxX,maxY;
 float angle[721];     /* vector of x axis data, degrees */
 char tempstr[20];
 char xstr[20], choice;
 FILE *out;
 for(i=0;i<=720;i++)
  {
  angle[i]=i/2.;
  }

 out= fopen("array.dat","a+");       /* open array.dat for appending */
/*************************************************************************
*    OUTPUT DATA TO FILE ARRAY.DAT CREATED/OPENED ON DEFAULT DRIVE      *
*************************************************************************/
 fprintf(out,"\t\t\tARRAY OUTPUT DATA FILE\n\n");
 fprintf(out,"Array Design Characteristics:\n");
 fprintf(out,"\tFrequency:\t\t%6.0f Hz\n",f);
 fprintf(out,"\tcelerity:\t\t%4.0f fps\n",c);
 fprintf(out,"\t# elements:\t\t%d\n ",m);
 fprintf(out,"\tSpacing:\t\t%4.1f ft\n",d);
 fprintf(out,"\twavelength:\t\t%3.1f ft\n",lambda);
 switch(wint)
```

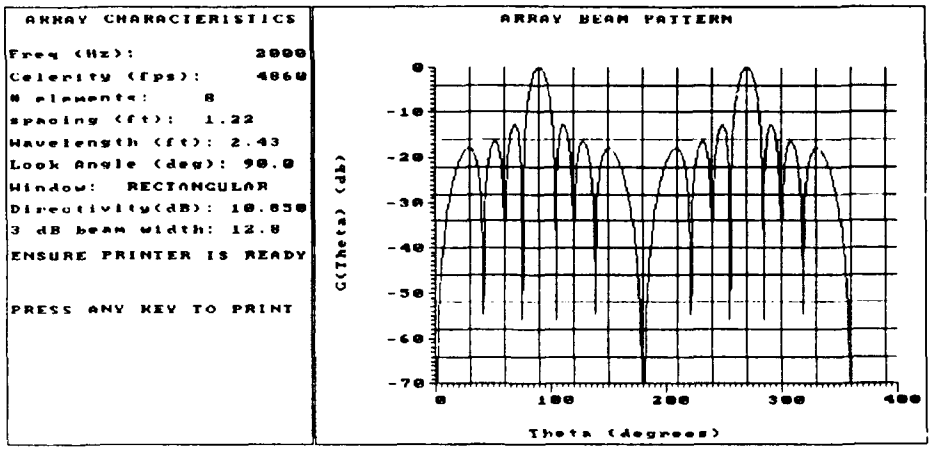
```
 {
 case 1:
  fprintf(out,"\tWindow:\t\tRECT\n");
  break;
 case 2:
  fprintf(out,"\tWindow:\t\t\tUSER\n");
  break;
 case 3:
  fprintf(out,"\tWindow:\t\t\tCOSINE\n");
  break;
 case 4:
  fprintf(out,"\tWindow:\t\t\tCOSINE SQUARED\n");
  break;
 case 5:
  fprintf(out,"\tWindow:\t\t\tTAYLOR\n");
  break;
 case 6:
  fprintf(out,"\tWindow:\t\t\tCHEBYSHEV\n");
  break;
 }
fprintf(out,"\tDirectivity:\t\t%5.3f dB\n",d);
fprintf(out,"\tBeamwidth:\t\t%4.2f degrees\n\n",bw);
if(wint != 1)
 {
 fprintf(out,"ARRAY WINDOW WEIGHTS\n\n");
 for(i=1;i<=m;i++)
     fprintf(out,"\tElement[%d] Amplitude Weight: %f\n",i,win[i]);
 }
fprintf(out,"\fNORMALIZED DIRECTIONAL GAIN PATTERN VALUES IN dB\n\n");
for(i=1;i<=360;i=i+4)
 fprintf(out,"%d deg: %4.2f\t%d deg: %4.2f\t%d deg: %4.2f\t%d deg: %4.2f\n",
 i,G[2*i],i+1,G[2*(i+1)],i+2,G[2*(i+2)],i+3,G[2*(i+3)]);
fprintf(out,"\f");
fclose(out);
/*****************************************************************************
*     PRODUCE PRINTER OUTPUT OF NORMALIZED DIRECTIONAL GAIN          *
*****************************************************************************/
   GetMaxCoords(&maxX,&maxY);
   DefGraphWindow(0,0,Round(maxX/3.),maxY,5);
   DefGraphWindow(Round(maxX/3),0,maxX,maxY,6);
   SetWin2PlotRatio(5,.2,0.14,0.05,0.14);
   SetWin2PlotRatio(6,.2,0.14 0.05,0.14);
   SetCurrentWindow(6);
   ClearWindow();
   SetAxesType(0,0);
   SelectColor(15);
   AutoAxes(angle,G,720,1);
   TitleWindow("ARRAY BEAM PATTERN");
   TitleXAxis("Theta (degrees)");
   TitleYAxis("G(Theta) (db)");
   BorderCurrentWindow(4);
   SelectColor(4);
   DrawGrid(6);
   LinePlotData(angle,G,721,15,0);  /* command to plot array #1 G(theta) */
   SetCurrentWindow(5);
   ClearWindow();
   BorderCurrentWindow(4);
   SelectColor(15);
   setlinestyleXX(0,0,1);
   TitleWindow("ARRAY CHARACTERISTICS");
   settextjustifyXX(0,0);
   sprintf(xstr,"%6.0f",f);
```

```
strcpy(tempstr,"Freq (Hz):      ");
strcat(tempstr, xstr );
LabelGraphWindow(20,880,tempstr,0,0);
sprintf(xstr,"%6.0f",c);
strcpy(tempstr,"Celerity (fps): ");
strcat(tempstr, xstr );
LabelGraphWindow(20.830,tempstr,0,0);
sprintf(xstr,"%3d",m);
strcpy(tempstr,"# elements: ");
strcat(tempstr, xstr );
LabelGraphWindow(20,780,tempstr,0,0);
sprintf(xstr,"%5.2f",d);
strcpy(tempstr,"spacing (ft): ");
strcat(tempstr, xstr );
LabelGraphWindow(20,730,tempstr,0,0);
sprintf(xstr,"%4.2f",lambda);
strcpy(tempstr,"Wavelength (ft): ");
strcat(tempstr, xstr );
LabelGraphWindow(20,680,tempstr,0,0);
sprintf(xstr,"%4.1f",theta);
strcpy(tempstr,"Look Angle (deg): ");
strcat(tempstr, xstr );
LabelGraphWindow(20,630,tempstr,0,0);
switch(wint)
 {
 case 1:
    strcpy(tempstr,"Window: RECTANGULAR");
    LabelGraphWindow(20,580,tempstr,0,0);
    break;
 case 2:
    strcpy(tempstr,"Window: USER");
    LabelGraphWindow(20,580,tempstr,0,0);
    break;
 case 3:
    strcpy(tempstr,"Window: COSINE");
    LabelGraphWindow(20,580,tempstr,0,0);
    break;
 case 4:
    strcpy(tempstr,"Window: COSINE SQUARED");
    LabelGraphWindow(20,580,tempstr,0,0);
    break;
 case 5:
    strcpy(tempstr,"Window: TAYLOR");
    LabelGraphWindow(20,580,tempstr,0,0);
    break;
 case 6:
    strcpy(tempstr,"Window: CHEBYSHEV");
    LabelGraphWindow(20,580,tempstr,0,0);
    break;
 }
sprintf(xstr,"%5.3f",di);
strcpy(tempstr,"Directivity(dB): ");
strcat(tempstr, xstr );
LabelGraphWindow(20,530,tempstr,0,0);
sprintf(xstr,"%2.1f",bw);
strcpy(tempstr,"3 dB beam width: ");
strcat(tempstr, xstr );
LabelGraphWindow(20,480,tempstr,0,0);
strcpy(tempstr,"ENSURE PRINTER IS READY");
LabelGraphWindow(20,420,tempstr,0,0);
```

```
    LabelGraphWindow(20,300,"PRESS ANY KEY TO PRINT",0,0);
    choice = getch();
    ScreenDump((int) prn,0,5,1.1,1.0,0,1,0,&error);
}
```

Appendix B ARRAY Output During Synthesis

Array Characteristics

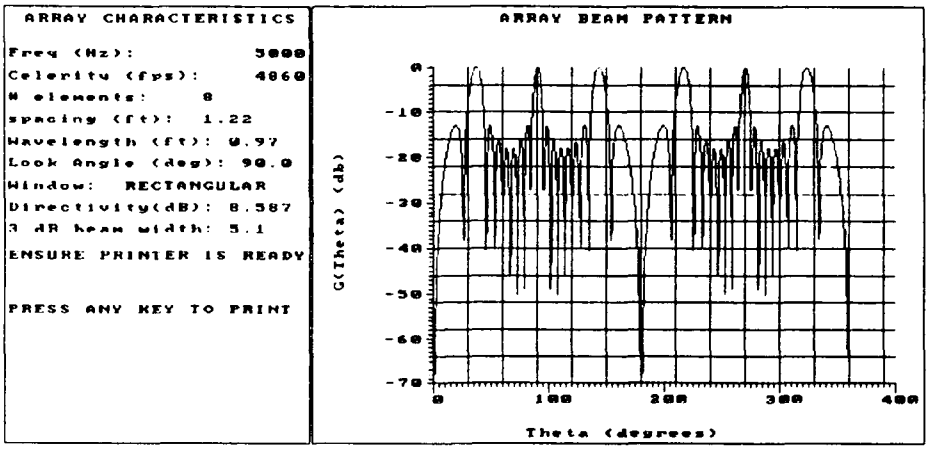


Frequency (Hz):	2000
Sound Speed(fps):	4860
# Elements:	8
Element Spacing(ft):	1.2
Wavelength (ft):	2.4
Directivity (dB):	1.215
Beamwidth (degrees):	12.77

Degrees	G[θ]	Degrees	G[θ]	Degrees	G[θ]	Degrees	Gθ]
1	-70	2	-60.38	3	-53.34	4	-48.35
5	-44.47	6	-41.31	7	-38.64	8	-36.33
9	-34.3	10	-32.5	11	-30.87	12	-29.39
13	-28.05	14	-26.81	15	-25.68	16	-24.64
17	-23.69	18	-22.81	19	-22.01	20	-21.28
21	-20.62	22	-20.02	23	-19.5	24	-19.04
25	-18.66	26	-18.35	27	-18.11	28	-17.96
29	-17.89	30	-17.92	31	-18.06	32	-18.32
33	-18.72	34	-19.28	35	-20.05	36	-21.07
37	-22.44	38	-24.32	39	-27.03	40	-31.43
41	-41.96	42	-38.64	43	-29.95	44	-25.69
45	-22.9	46	-20.89	47	-19.38	48	-18.24
49	-17.41	50	-16.83	51	-16.51	52	-16.43
53	-16.61	54	-17.08	55	-17.9	56	-19.17
57	-21.1	58	-24.14	59	-29.78	60	-70
61	-29.28	62	-23.15	63	-19.62	64	-17.23
65	-15.51	66	-14.28	67	-13.44	68	-12.95
69	-12.8	70	-13.01	71	-13.65	72	-14.85
73	-16.88	74	-20.51	75	-29.13	76	-29.33
77	-19.03	78	-14.13	79	-10.85	80	-8.41
81	-6.48	82	-4.92	83	-3.66	84	-2.62
85	-1.78	86	-1.12	87	-0.63	88	-0.28
89	-0.07	90	0	91	-0.07	92	-0.28
93	-0.63	94	-1.12	95	-1.78	96	-2.62
97	-3.66	98	-4.92	99	-6.48	100	-8.41
101	-10.85	102	-14.13	103	-19.03	104	-29.33
105	-29.13	106	-20.51	107	-16.88	108	-14.85
109	-13.65	110	-13.01	111	-12.8	112	-12.95
113	-13.44	114	-14.28	115	-15.51	116	-17.23
117	-19.62	118	-23.15	119	-29.28	120	-70
121	-29.78	122	-24.14	123	-21.1	124	-19.17
125	-17.9	126	-17.08	127	-16.61	128	-16.43
129	-16.51	130	-16.83	131	-17.41	132	-18.24

133	-19.38	134	-20.89	135	-22.9	136	-25.69
137	-29.95	138	-38.64	139	-41.96	140	-31.43
141	-27.03	142	-24.32	143	-22.44	144	-21.07
145	-20.05	146	-19.28	147	-18.72	148	-18.32
149	-18.06	150	-17.92	151	-17.89	152	-17.96
153	-18.11	154	-18.35	155	-18.66	156	-19.04
157	-19.5	158	-20.02	159	-20.62	160	-21.28
161	-22.01	162	-22.81	163	-23.69	164	-24.64
165	-25.68	166	-26.81	167	-28.05	168	-29.39
169	-30.87	170	-32.5	171	-34.3	172	-36.33
173	-38.64	174	-41.31	175	-44.47	176	-48.35
177	-53.34	178	-60.38	179	-70	180	-70

Array Characteristics

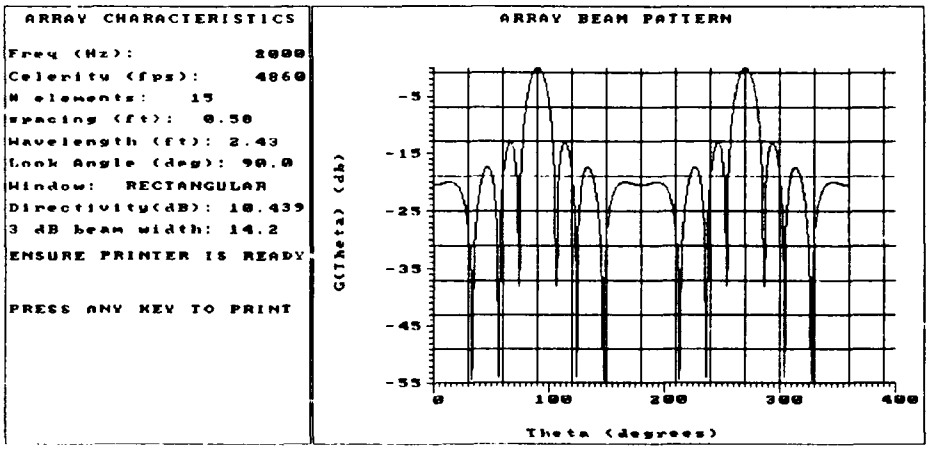


Frequency (Hz): 5000
Sound Speed(fps): 4860
Elements: 8
Eiement Spacing(ft): 1.2
Wavelength (ft): 1
Directivity (dB): 1.215
Beamwidth (degrees): 5.09

Degrees	G[θ]	Degrees	G[θ]	Degrees	G[θ]	Degrees	Gθ]
1	-61.45	2	-49.39	3	-42.33	4	-37.3
5	-33.39	6	-30.19	7	-27.48	8	-25.14
9	-23.08	10	-21.26	11	-19.64	12	-18.2
13	-16.92	14	-15.81	15	-14.85	16	-14.06
17	-13.44	18	-13.02	19	-12.81	20	-12.87
21	-13.27	22	-14.12	23	-15.67	24	-18.44
25	-24.34	26	-38.05	27	-20.03	28	-14.01
29	-10.18	30	-7.39	31	-5.23	32	-3.54
33	-2.22	34	-1.22	35	-0.52	36	-0.11
37	0	38	-0.2	39	-0.75	40	-1.69
41	-3.11	42	-5.17	43	-8.15	44	-12.8
45	-22.21	46	-25.69	47	-16.49	48	-13.57
49	-12.8	50	-13.54	51	-15.9	52	-21
53	-39.87	54	-23.98	55	-18.5	56	-16.61
57	-16.7	58	-18.74	59	-23.98	60	-70
61	-24.33	62	-19.47	63	-17.93	64	-18.56
65	-21.74	66	-31.61	67	-28.9	68	-20.92
69	-18.25	70	-18.11	71	-20.54	72	-28.57
73	-29.75	74	-20.17	75	-16.99	76	-16.55
77	-18.85	78	-27.49	79	-25.64	80	-16.71
81	-13.45	82	-12.89	83	-15.28	84	-27.05
85	-16.7	86	-8.5	87	-4.28	88	-1.79
89	-0.43	90	0	91	-0.43	92	-1.79
93	-4.28	94	-8.5	95	-16.7	96	-27.05
97	-15.28	98	-12.89	99	-13.45	100	-16.71
101	-25.64	102	-27.49	103	-18.85	104	-16.55
105	-16.99	106	-20.17	107	-29.75	108	-28.57
109	-20.54	110	-18.11	111	-18.25	112	-20.92
113	-28.9	114	-31.61	115	-21.74	116	-18.56
117	-17.93	118	-19.47	119	-24.33	120	-70
121	-23.98	122	-18.74	123	-16.7	124	-16.61
125	-18.5	126	-23.98	127	-39.87	128	-21

129	-15.9	130	-13.54	131	-12.8	132	-13.57
133	-16.49	134	-25.69	135	-22.21	136	-12.8
137	-8.15	138	-5.17	139	-3.11	140	-1.69
141	-0.75	142	-0.2	143	0	144	-0.11
145	-0.52	146	-1.22	147	-2.22	148	-3.54
149	-5.23	150	-7.39	151	-10.18	152	-14.01
153	-20.03	154	-38.05	155	-24.34	156	-18.44
157	-15.67	158	-14.12	159	-13.27	160	-12.87
161	-12.81	162	-13.02	163	-13.44	164	-14.06
165	-14.85	166	-15.81	167	-16.92	168	-18.2
169	-19.64	170	-21.26	171	-23.08	172	-25.14
173	-27.48	174	-30.19	175	-33.39	176	-37.3
177	-42.33	178	-49.39	179	-61.45	180	-70

Array Characteristics

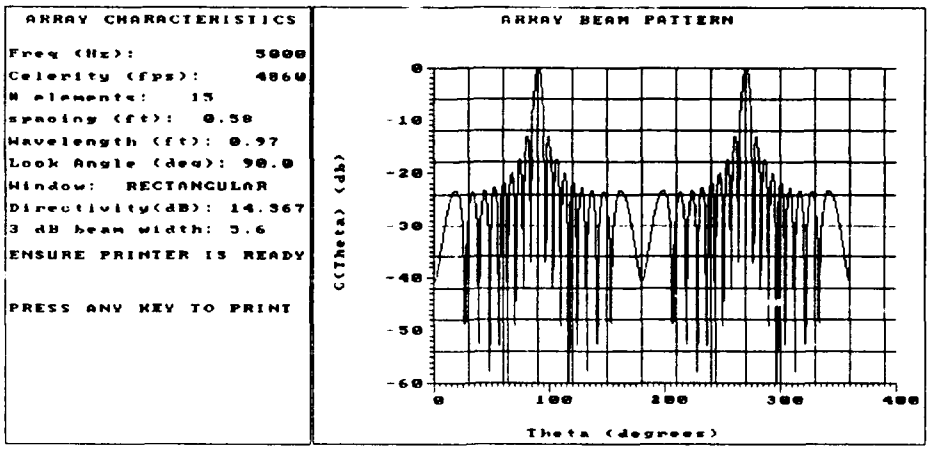


Frequency (Hz):	2000
Sound Speed(fps):	4860
# Elements:	15
Element Spacing(ft):	0.6
Wavelength (ft):	2.4
Directivity (dB):	0.581
Beamwidth (degrees):	14.2

Degrees	G[θ]	Degrees	G[θ]	Degrees	G[θ]	Degrees	Gθ]
1	-20.49	2	-20.48	3	-20.45	4	-20.42
5	-20.38	6	-20.33	7	-20.28	8	-20.23
9	-20.18	10	-20.13	11	-20.08	12	-20.04
13	-20.02	14	-20.01	15	-20.02	16	-20.05
17	-20.11	18	-20.21	19	-20.35	20	-20.53
21	-20.78	22	-21.09	23	-21.48	24	-21.97
25	-22 57	26	-23.32	27	-24.26	28	-25.46
29	-27	30	-29.09	31	-32.12	32	-37.3
33	-54.6	34	-39.57	35	-32.64	36	-28.76
37	-26.07	38	-24.03	39	-22.41	40	-21.11
41	-20.04	42	-19.19	43	-18.51	44	-18
45	-17.66	46	-17.47	47	-17.46	48	-17.63
49	-18	50	-18.63	51	-19.56	52	-20.89
53	-22.84	54	-25.82	55	-31.17	56	-54.24
57	-31.91	58	-25.39	59	-21.67	60	-19.11
61	-17.22	62	-15.79	63	-14.71	64	-13.93
65	-13.41	66	-13.16	67	-13.18	68	-13.5
69	-14.18	70	-15.35	71	-17.22	72	-20.36
73	-26.8	74	-38.03	75	-22.15	76	-16.48
77	-12.88	78	-10.23	79	-8.15	80	-6.45
81	-5.05	82	-3.88	83	-2.91	84	-2.1
85	-1.44	86	-0.91	87	-0.51	88	-0.22
89	-0.06	90	0	91	-0.06	92	-0.22
93	-0.51	94	-0.91	95	-1.44	96	-2.1
97	-2.91	98	-3.88	99	-5.05	100	-6.45
101	-8.15	102	-10.23	103	-12.88	104	-16.48
105	-22.15	106	-38.03	107	-26.8	108	-20.36
109	-17.22	110	-15.35	111	-14.18	112	-13.5
113	-13.18	114	-13.16	115	-13.41	116	-13.93
117	-14.71	118	-15.79	119	-17.22	120	-19.11
121	-21.67	122	-25.39	123	-31.91	124	-54 24
125	-31 17	126	-25.82	127	-22.84	128	-20.89
129	-19.56	130	-18.63	131	-18	132	-17.63

133	-17.46	134	-17.47	135	-17.66	136	-18
137	-18.51	138	-19.19	139	-20.04	140	-21.11
141	-22.41	142	-24.03	143	-26.07	144	-28.76
145	-32.64	146	-39.57	147	-54.6	148	-37.3
149	-32.12	150	-29.09	151	-27	152	-25.46
153	-24.26	154	-23.32	155	-22.57	156	-21.97
157	-21.48	158	-21.09	159	-20.78	160	-20.53
161	-20.35	162	-20.21	163	-20.11	164	-20.05
165	-20.02	166	-20.01	167	-20.02	168	-20.04
169	-20.08	170	-20.13	171	-20.18	172	-20.23
173	-20.28	174	-20.33	175	-20.38	176	-20.42
177	-20.45	178	-20.48	179	-20.49	180	-20.5

Array Characteristics

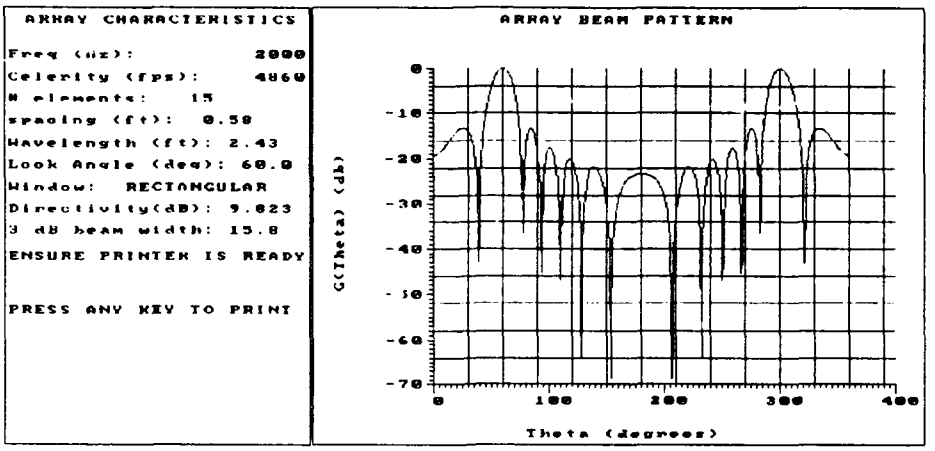


Frequency (Hz):	5000
Sound Speed(fps):	4860
# Elements:	15
Element Spacing(ft):	0.6
Wavelength (ft):	1
Directivity (dB):	0.581
Beamwidth (degrees):	5.64

Degrees	G[θ]	Degrees	G[θ]	Degrees	G[θ]	Degrees	Gθ]
1	-40.52	2	-39.74	3	-38.58	4	-37.18
5	-35.68	6	-34.16	7	-32.67	8	-31.25
9	-29.93	10	-28.7	11	-27.58	12	-26.58
13	-25.7	14	-24.94	15	-24.31	16	-23.83
17	-23.5	18	-23.34	19	-23.38	20	-23.65
21	-24.21	22	-25.12	23	-26.53	24	-28.69
25	-32.2	26	-39.29	27	-48.4	28	-34.55
29	-29.61	30	-26.75	31	-24.96	32	-23.93
33	-23.53	34	-23.75	35	-24.69	36	-26.59
37	-30.12	38	-38.14	39	-41.93	40	-31.01
41	-26.74	42	-24.49	43	-23.46	44	-23.44
45	-24.49	46	-27.02	47	-32.58	48	-58.01
49	-31.5	50	-26.17	51	-23.68	52	-22.76
53	-23.16	54	-25.14	55	-29.93	56	-52.84
57	-30.76	58	-24.92	59	-22.4	60	-21.67
61	-22.57	62	-25.67	63	-34.65	64	-33.7
65	-24.69	66	-21.23	67	-20.03	68	-20.65
69	-23.65	70	-33.07	71	-30.42	72	-21.83
73	-18.49	74	-17.44	75	-18.4	76	-22.36
77	-42.2	78	-22.69	79	-16.44	80	-13.76
81	-13.17	82	-14.86	83	-21.56	84	-23.36
85	-11.69	86	-6.52	87	-3.39	88	-1.44
89	-0.35	90	0	91	-0.35	92	-1.44
93	-3.39	94	-6.52	95	-11.69	96	-23.36
97	-21.56	98	-14.86	99	-13.17	100	-13.76
101	-16.44	102	-22.69	103	-42.2	104	-22.36
105	-18.4	106	-17.44	107	-18.49	108	-21.83
109	-30.42	110	-33.07	111	-23.65	112	-20.65
113	-20.03	114	-21.23	115	-24.69	116	-33.7
117	-34.65	118	-25.67	119	-22.57	120	-21.67
121	-22.4	122	-24.92	123	-30.76	124	-52.84
125	-29.93	126	-25.14	127	-23.16	128	-22.76
129	-23.68	130	-26.17	131	-31.5	132	-58.01

133	-32.58	134	-27.02	135	-24.49	136	-23.44
137	-23.46	138	-24.49	139	-26.74	140	-31.01
141	-41.93	142	-38.14	143	-30.12	144	-26.59
145	-24.69	146	-23.75	147	-23.53	148	-23.93
149	-24.96	150	-26.75	151	-29.61	152	-34.55
153	-48.4	154	-39.29	155	-32.2	156	-28.69
157	-26.53	158	-25.12	159	-24.21	160	-23.65
161	-23.38	162	-23.34	163	-23.5	164	-23.83
165	-24.31	166	-24.94	167	-25.7	168	-26.58
169	-27.58	170	-28.7	171	-29.93	172	-31.25
173	-32.67	174	-34.16	175	-35.68	176	-37.18
177	-38.58	178	-39.74	179	-40.52	180	-40.8

Array Characteristics

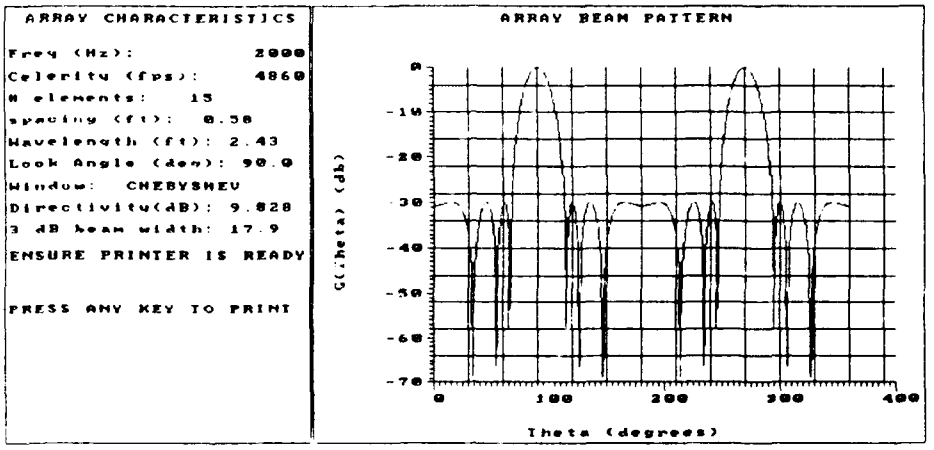


Frequency (Hz):	2000
Sound Speed(fps):	4860
# Elements:	15
Element Spacing(ft):	0.6
Wavelength (ft):	2.4
Directivity (dB):	0.581
Beamwidth (degrees):	15.78

Degrees	G[θ]	Degrees	G[θ]	Degrees	G[θ]	Degrees	Gθ]
1	-19.09	2	-19.03	3	-18.92	4	-18.77
5	-18.59	6	-18.37	7	-18.12	8	-17.84
9	-17.54	10	-17.22	11	-16.89	12	-16.55
13	-16.2	14	-15.86	15	-15.51	16	-15.18
17	-14.85	18	-14.54	19	-14.25	20	-13.98
21	-13.74	22	-13.53	23	-13.36	24	-13.23
25	-13.15	26	-13.13	27	-13.17	28	-13.29
29	-13.5	30	-13.81	31	-14.25	32	-14.85
33	-15.66	34	-16.75	35	-18.23	36	-20.36
37	-23.67	38	-30.11	39	-43.11	40	-26.18
41	-20.44	42	-16.8	43	-14.09	44	-11.92
45	-10.12	46	-8.58	47	-7.24	48	-6.06
49	-5.03	50	-4.11	51	-3.31	52	-2.6
53	-1.98	54	-1.46	55	-1.01	56	-0.65
57	-0.37	58	-0.16	59	-0.04	60	0
61	-0.04	62	-0.17	63	-0.39	64	-0.71
65	-1.12	66	-1.65	67	-2.3	68	-3.09
69	-4.04	70	-5.17	71	-6.52	72	-8.17
73	-10.19	74	-12.77	75	-16.29	76	-21.82
77	-36.64	78	-26.98	79	-20.32	80	-17.11
81	-15.21	82	-14.05	83	-13.4	84	-13.14
85	-13.22	86	-13.63	87	-14.36	88	-15.45
89	-16.99	90	-19.11	91	-22.17	92	-27.11
93	-39.33	94	-33.57	95	-25.99	96	-22.4
97	-20.23	98	-18.84	99	-17.99	100	-17.55
101	-17.45	102	-17.67	103	-18.22	104	-19.11
105	-20.4	106	-22.22	107	-24.8	108	-28.76
109	-36.48	110	-44.28	111	-31.64	112	-27
113	-24.3	114	-22.53	115	-21.35	116	-20.58
117	-20.15	118	-20.01	119	-20.13	120	-20.5
121	-21.13	122	-22.06	123	-23.33	124	-25.04
125	-27.38	126	-30.8	127	-36.66	128	-64.1
129	-37.72	130	-31.73	131	-28.44	132	-26.27

133	-24.74	134	-23.62	135	-22.82	136	-22.26
137	-21.9	138	-21.71	139	-21.68	140	-21.79
141	-22.03	142	-22.41	143	-22.93	144	-23.58
145	-24.39	146	-25.37	147	-26.55	148	-27.98
149	-29.73	150	-31.94	151	-34.91	152	-39.36
153	-48.48	154	-50.51	155	-40.51	156	-36.17
157	-33.44	158	-31.49	159	-30	160	-28.83
161	-27.88	162	-27.1	163	-26.45	164	-25.91
165	-25.45	166	-25.07	167	-24.74	168	-24.47
169	-24.24	170	-24.04	171	-23.88	172	-23.75
173	-23.63	174	-23.54	175	-23.47	176	-23.41
177	-23.37	178	-23.34	179	-23.32	180	-23.32

Array Characteristics



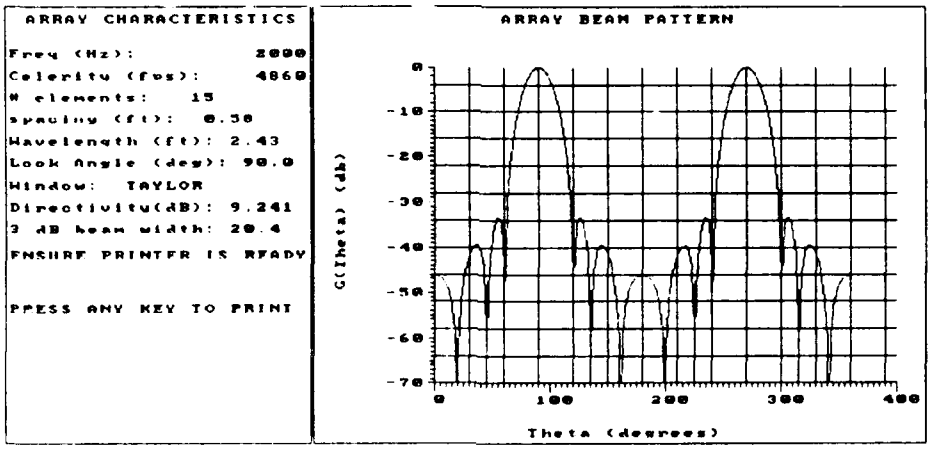
Frequency (Hz):	2000
Sound Speed(fps):	4860
# Elements:	15
Element Spacing(ft):	0.6
Wavelength (ft):	2.4
Directivity (dB):	
Beamwidth (degrees):	0.581
	17.89

Element	Amplitude Weight
1	0.281302
2	0.326266
3	0.474929
4	0.630153
5	0.775678
6	0.894747
7	0.972813
8	0.5
9	0.972813
10	0.894747
11	0.775678
12	0.630153
13	0.474929
14	0.326266
15	0.281302

Degrees	G[θ]	Degrees	G[θ]	Degrees	G[θ]	Degrees	Gθ]
1	-30.71	2	-30.69	3	-30.66	4	-30.62
5	-30.56	6	-30.5	7	-30.44	8	-30.36
9	-30.29	10	-30.22	11	-30.15	12	-30.09
13	-30.05	14	-30.01	15	-30	16	-30.01
17	-30.05	18	-30.12	19	-30.24	20	-30.4
21	-30.62	22	-30.91	23	-31.27	24	-31.73
25	-32.29	26	-33	27	-33.88	28	-34.98
29	-36.39	30	-38.25	31	-40.84	32	-44.86
33	-53.23	34	-56.52	35	-45.59	36	-40.87
37	-37.86	38	-35.68	39	-34.03	40	-32.74
41	-31.74	42	-30.99	43	-30.45	44	-30.12
45	-30	46	-30.09	47	-30.43	48	-31.03
49	-31.99	50	-33.39	51	-35.47	52	-38.72
53	-44.79	54	-66.67	55	-43.27	56	-37.6
57	-34.38	58	-32.29	59	-30.94	60	-30.19

61	-30.01	62	-30.51	63	-31.96	64	-35.18
65	-44.5	66	-40.44	67	-30.67	68	-25.42
69	-21.67	70	-18.69	71	-16.21	72	-14.08
73	-12.22	74	-10.58	75	-9.11	76	-7.8
77	-6.62	78	-5.57	79	-4.63	80	-3.78
81	-3.04	82	-2.38	83	-1.81	84	-1.32
85	-0.91	86	-0.58	87	-0.33	88	-0.15
89	-0.04	90	0	91	-0.04	92	-0.15
93	-0.33	94	-0.58	95	-0.91	96	-1.32
97	-1.81	98	-2.38	99	-3.04	100	-3.78
101	-4.63	102	-5.57	103	-6.62	104	-7.8
105	-9.11	106	-10.58	107	-12.22	108	-14.08
109	-16.21	110	-18.69	111	-21.67	112	-25.42
113	-30.67	114	-40.44	115	-44.5	116	-35.18
117	-31.96	118	-30.51	119	-30.01	120	-30.19
121	-30.94	122	-32.29	123	-34.38	124	-37.6
125	-43.27	126	-66.67	127	-44.79	128	-38.72
129	-35.47	130	-33.39	131	-31.99	132	-31.03
133	-30.43	134	-30.09	135	-30	136	-30.12
137	-30.45	138	-30.99	139	-31.74	140	-32.74
141	-34.03	142	-35.68	143	-37.86	144	-40.87
145	-45.59	146	-56.52	147	-53.23	148	-44.86
149	-40.84	150	-38.25	151	-36.39	152	-34.98
153	-33.88	154	-33	155	-32.29	156	-31.73
157	-31.27	158	-30.91	159	-30.62	160	-30.4
161	-30.24	162	-30.12	163	-30.05	164	-30.01
165	-30	166	-30.01	167	-30.05	168	-30.09
169	-30.15	170	-30.22	171	-30.29	172	-30.36
173	-30.44	174	-30.5	175	-30.56	176	-30.62
177	-30.66	178	-30.69	179	-30.71	180	-30.72

Array Characteristics



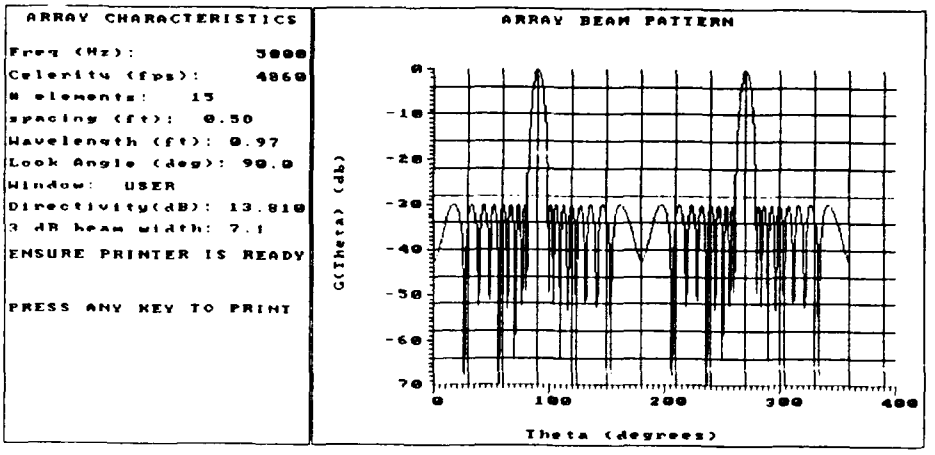
Frequency (Hz):	2000
Sound Speed(fps):	4860
# Elements:	15
Element Spacing(ft):	0.6
Wavelength (ft):	2.4
Directivity (dB):	0.581
Beamwidth (degrees):	20.44

Element	Amplitude Weight
1	1
2	2.457965
3	4.369205
4	6.542757
5	8.704096
6	10.54437
7	11.78125
8	6.10859
9	11.78125
10	10.54437
11	8.704096
12	6.542757
13	4.369205
14	2.457965
15	1

Degrees	G[θ]	Degrees	G[θ]	Degrees	G[θ]	Degrees	Gθ]
1	-46.22	2	-46.27	3	-46.36	4	-46.5
5	-46.67	6	-46.9	7	-47.18	8	-47.52
9	-47.94	10	-48.46	11	-49.08	12	-49.84
13	-50.79	14	-51.98	15	-53.54	16	-55.64
17	-58.75	18	-64.23	19	-70	20	-64.48
21	-57.95	22	-54.1	23	-51.32	24	-49.15
25	-47.36	26	-45.86	27	-44.58	28	-43.47
29	-42.51	30	-41.7	31	-41.01	32	-40.44
33	-40	34	-39.68	35	-39.5	36	-39.47
37	-39.6	38	-39.92	39	-40.49	40	-41.35
41	-42.64	42	-44.56	43	-47.61	44	-53.44
45	-70	46	-51.44	47	-45.39	48	-41.74
49	-39.15	50	-37.2	51	-35.71	52	-34.6
53	-33.81	54	-33.37	55	-33.29	56	-33.66
57	-34.64	58	-36.64	59	-40.82	60	-57.16
61	-41.26	62	-33.45	63	-28.72	64	-25.19

65	-22.33	66	-19.89	67	-17.76	68	-15.87
69	-14.17	70	-12.63	71	-11.22	72	-9.93
73	-8.75	74	-7.67	75	-6.68	76	-5.77
77	-4.93	78	-4.18	79	-3.49	80	-2.87
81	-2.31	82	-1.82	83	-1.39	84	-1.02
85	-0.7	86	-0.45	87	-0.25	88	-0.11
89	-0.03	90	0	91	-0.03	92	-0.11
93	-0.25	94	-0.45	95	-0.7	96	-1.02
97	-1.39	98	-1.82	99	-2.31	100	-2.87
101	-3.49	102	-4.18	103	-4.93	104	-5.77
105	-6.68	106	-7.67	107	-8.75	108	-9.93
109	-11.22	110	-12.63	111	-14.17	112	-15.87
113	-17.76	114	-19.89	115	-22.33	116	-25.19
117	-28.72	118	-33.45	119	-41.26	120	-57.16
121	-40.82	122	-36.64	123	-34.64	124	-33.66
125	-33.29	126	-33.37	127	-33.81	128	-34.6
129	-35.71	130	-37.2	131	-39.15	132	-41.74
133	-45.39	134	-51.44	135	-70	136	-53.44
137	-47.61	138	-44.56	139	-42.64	140	-41.35
141	-40.49	142	-39.92	143	-39.6	144	-39.47
145	-39.5	146	-39.68	147	-40	148	-40.44
149	-41.01	150	-41.7	151	-42.51	152	-43.47
153	-44.58	154	-45.86	155	-47.36	156	-49.15
157	-51.32	158	-54.1	159	-57.95	160	-64.48
161	-70	162	-64.23	163	-58.75	164	-55.64
165	-53.54	166	-51.98	167	-50.79	168	-49.84
169	-49.08	170	-48.46	171	-47.94	172	-47.52
173	-47.18	174	-46.9	175	-46.67	176	-46.5
177	-46.36	178	-46.27	179	-46.22	180	-46.2

Array Characteristics



Frequency (Hz):	5000
Sound Speed(fps):	4860
# Elements:	15
Element Spacing(ft):	0.6
Wavelength (ft):	1
Directivity (dB):	0.581
Beamwidth (degrees):	7.12

Element	Amplitude Weight
1	0.281302
2	0.326266
3	0.474929
4	0.630153
5	0.775678
6	0.894747
7	0.972813
8	0.5
9	0.972813
10	0.894747
11	0.775678
12	0.630153
13	0.474929
14	0.326266
15	0.281302

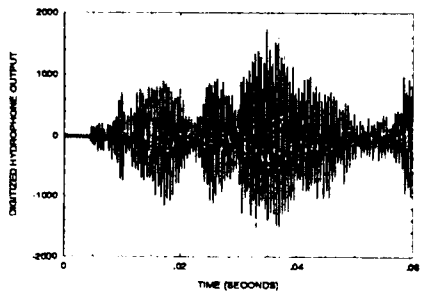
Degrees	G[θ]	Degrees	G[θ]	Degrees	G[θ]	Degrees	Gθ]
1	-42.32	2	-41.9	3	-41.23	4	-40.39
5	-39.42	6	-38.38	7	-37.31	8	-36.26
9	-35.24	10	-34.27	11	-33.37	12	-32.55
13	-31.82	14	-31.2	15	-30.69	16	-30.31
17	-30.08	18	-30.01	19	-30.13	20	-30.48
21	-31.12	22	-32.14	23	-33.67	24	-36.02
25	-39.92	26	-48.53	27	-50.46	28	-40.15
29	-35.72	30	-33.06	31	-31.38	32	-30.39
33	-30.01	34	-30.22	35	-31.12	36	-32.91
37	-36.19	38	-43.18	39	-52.28	40	-38.58
41	-33.9	42	-31.45	43	-30.25	44	-30.04
45	-30.84	46	-32.91	47	-37.25	48	-50.5
49	-41.61	50	-34.57	51	-31.48	52	-30.16
53	-30.16	54	-31.57	55	-35.06	56	-44.36
57	-43.49	58	-34.66	59	-31.28	60	-30.04

61	-30.45	62	-32.73	63	-38.64	64	-53.59
65	-36.12	66	-31.64	67	-30.08	68	-30.51
69	-33.31	70	-41.62	71	-42.83	72	-33.45
73	-30.44	74	-30.21	75	-32.88	76	-43.04
77	-38.55	78	-31.3	79	-30.17	80	-36.65
81	-30.82	82	-19.7	83	-13.57	84	-9.32
85	-6.17	86	-3.82	87	-2.1	88	-0.92
89	-0.23	90	0	91	-0.23	92	-0.92
93	-2.1	94	-3.82	95	-6.17	96	-9.32
97	-13.57	98	-19.7	99	-30.82	100	-36.65
101	-30.17	102	-31.3	103	-38.55	104	-43.04
105	-32.88	106	-30.21	107	-30.44	108	-33.45
109	-42.83	110	-41.62	111	-33.31	112	-30.51
113	-30.08	114	-31.64	115	-36.12	116	-53.59
117	-38.64	118	-32.73	119	-30.45	120	-30.04
121	-31.28	122	-34.66	123	-43.49	124	-44.36
125	-35.06	126	-31.57	127	-30.16	128	-30.16
129	-31.48	130	-34.57	131	-41.61	132	-50.5
133	-37.25	134	-32.91	135	-30.84	136	-30.04
137	-30.25	138	-31.45	139	-33.9	140	-38.58
141	-52.28	142	-43.18	143	-36.19	144	-32.91
145	-31.12	146	-30.22	147	-30.01	148	-30.39
149	-31.38	150	-33.06	151	-35.72	152	-40.15
153	-50.46	154	-48.53	155	-39.92	156	-36.02
157	-33.67	158	-32.14	159	-31.12	160	-30.48
161	-30.13	162	-30.01	163	-30.08	164	-30.31
165	-30.69	166	-31.2	167	-31.82	168	-32.55
169	-33.37	170	-34.27	171	-35.24	172	-36.26
173	-37.31	174	-38.38	175	-39.42	176	-40.39
177	-41.23	178	-41.9	179	-42.32	180	-42.47

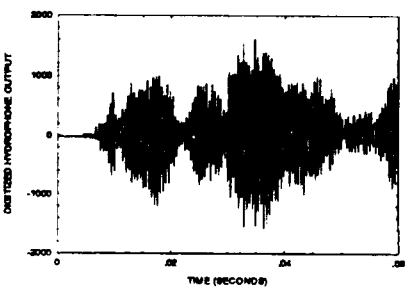
Appendix C Digitized Hydrophone Output

Pages 114 thru 132

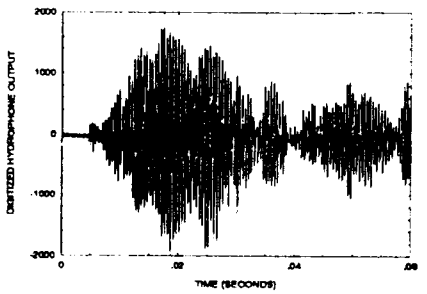
3.1 Appendix C1 Unaveraged, Untapered, Hydrophone Data



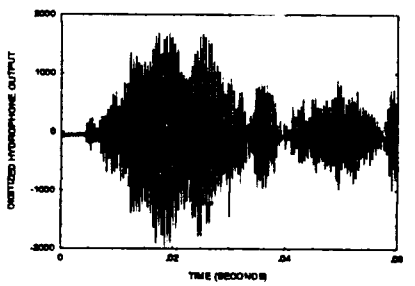
Hydrophone 1 with scattering plate



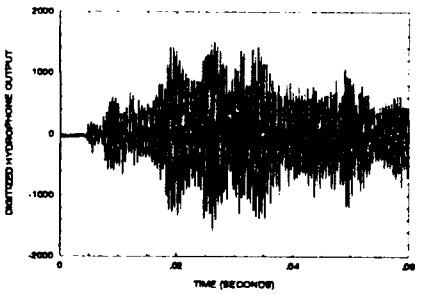
Hydrophone 1 without scattering plate



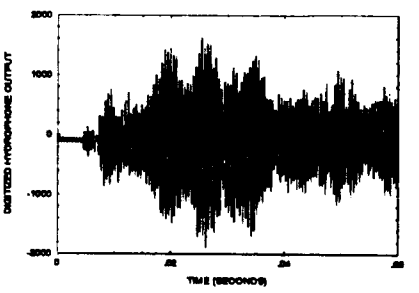
Hydrophone 2 with scattering plate



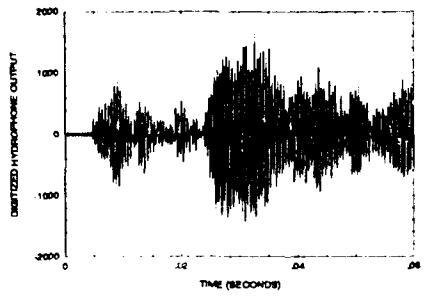
Hydrophone 2 without scattering plate



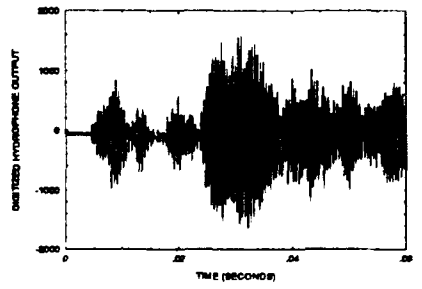
Hydrophone 3 with s attering plate



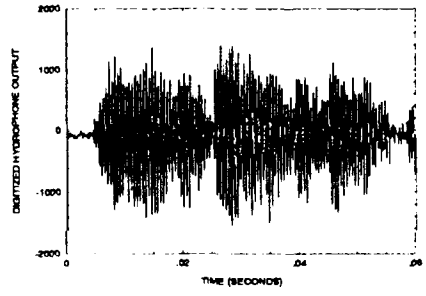
Hydrophone 3 without scattering plate



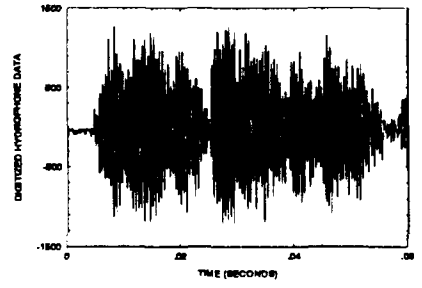
Hydrophone 4 with scattering plate



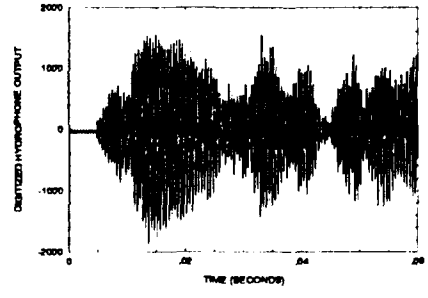
Hydrophone 4 without scattering plate



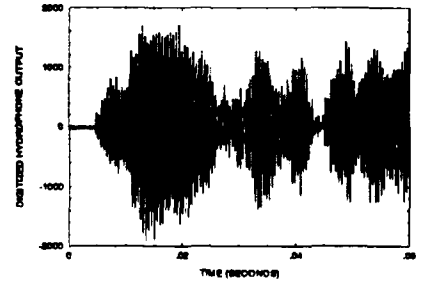
Hydrophone 5 with scattering plate



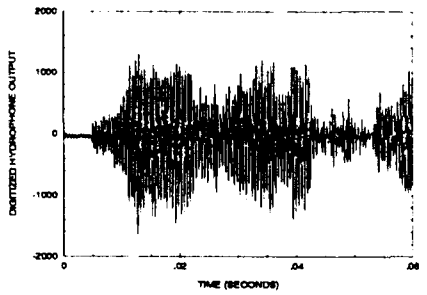
Hydrophone 5 without scattering plate



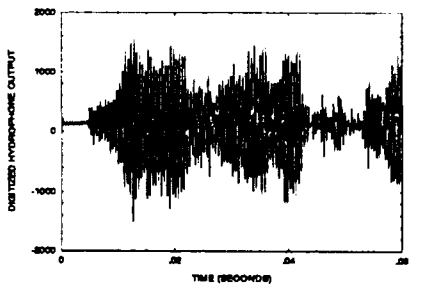
Hydrophone 6 with scattering plate



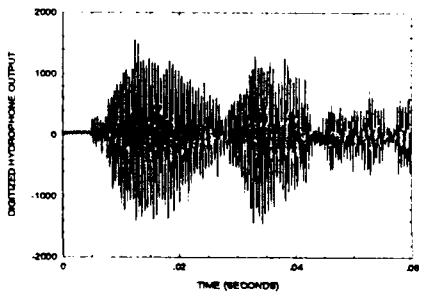
Hydrophone 6 without scattering plate



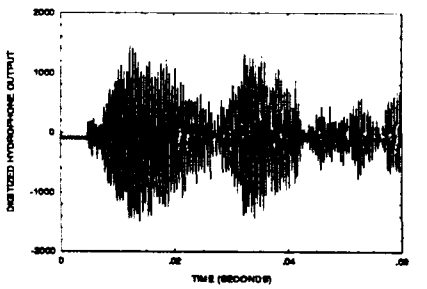
Hydrophone 7 with scattering plate



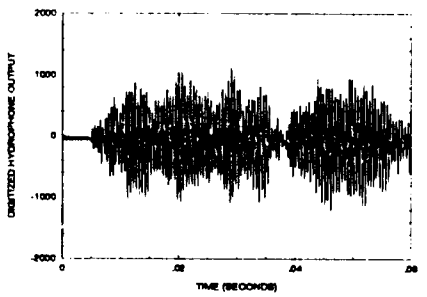
Hydrophone 7 without scattering plate



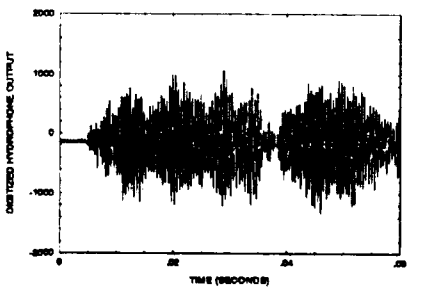
Hydrophone 8 with scattering plate



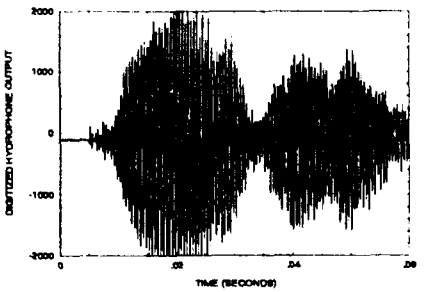
Hydrophone 8 without scattering plate



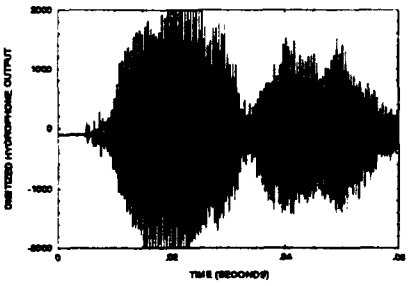
Hydrophone 9 with scattering plate



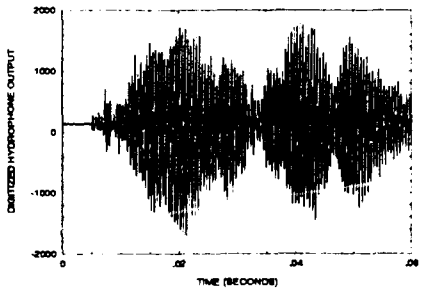
Hydrophone 9 without scattering plate



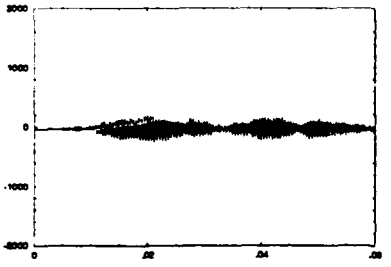
Hydrophone 10 with scattering plate



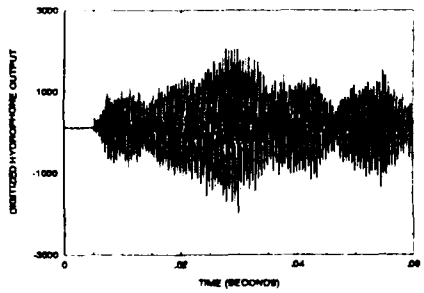
Hydrophone 10 without scattering plate



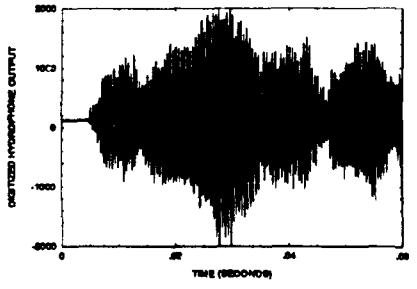
Hydrophone 11 with scattering plate



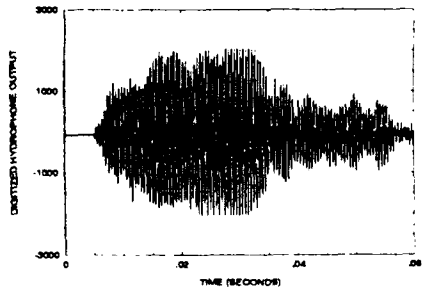
Hydrophone 11 without scattering plate



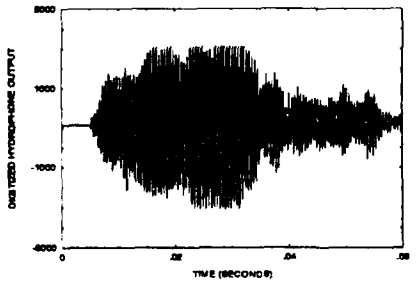
Hydrophone 12 with scattering plate



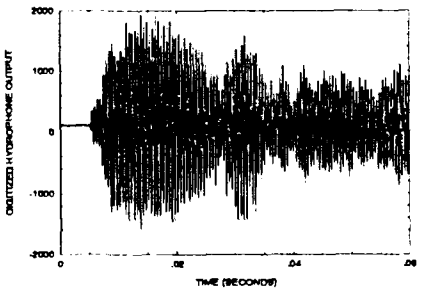
Hydrophone 12 without scattering plate



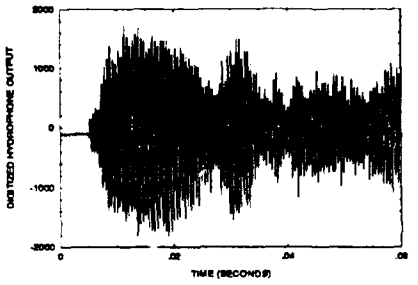
Hydrophone 13 with scattering plate



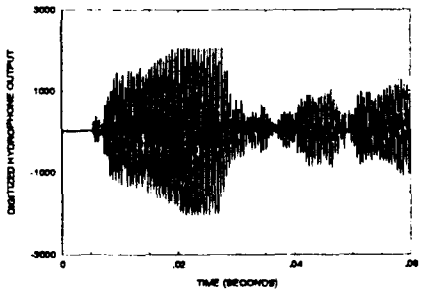
Hydrophone 13 without scattering plate



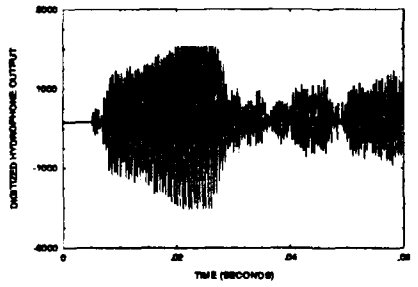
Hydrophone 14 with scattering plate



Hydrophone 14 without scattering plate

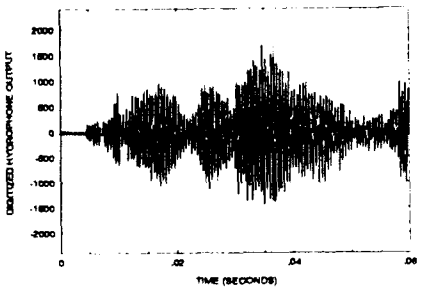


Hydrophone 15 with scattering plate

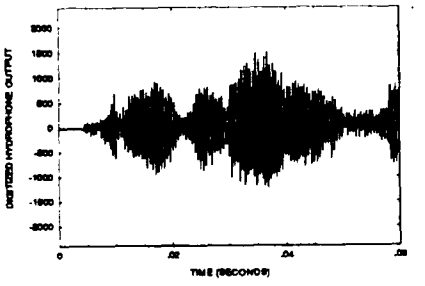


Hydrophone 15 without scattering plate

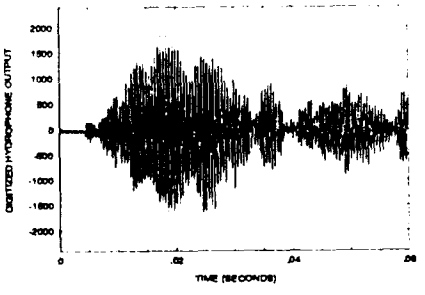
3.2 Appendix C2 20 Pulse, Averaged, Untapered, Hydrophone Data



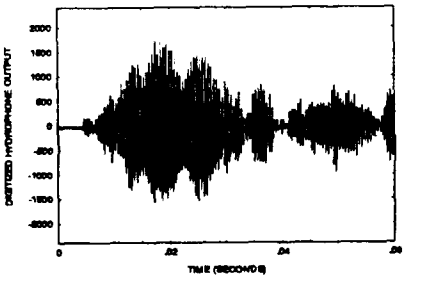
Hydrophone 1 with scattering plate



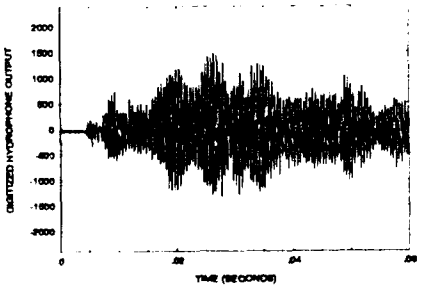
Hydrophone 1 without scattering plate



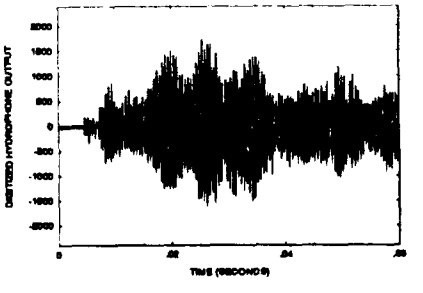
Hydrophone 2 with scattering plate



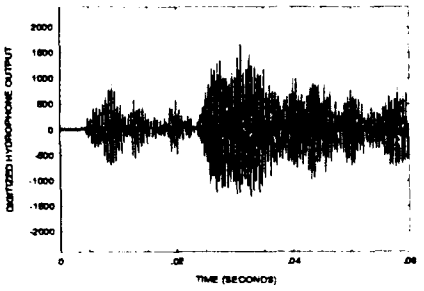
Hydrophone 2 without scattering plate



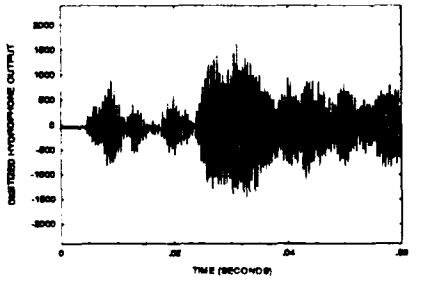
Hydrophone 3 with scattering plate



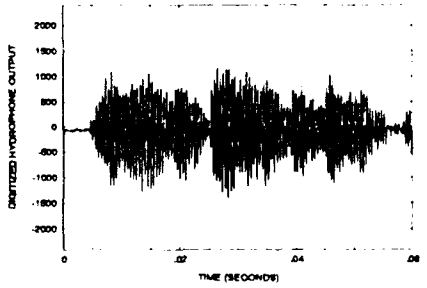
Hydrophone 3 without scattering plate



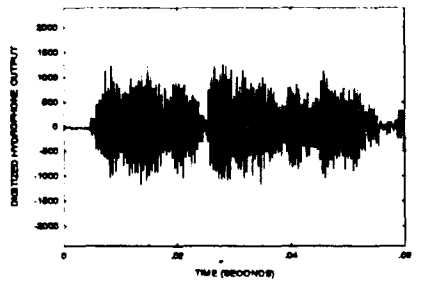
Hydrophone 4 with scattering plate



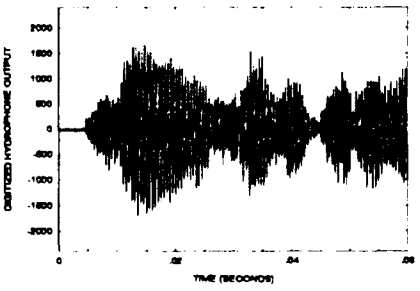
Hydrophone 4 without scattering plate



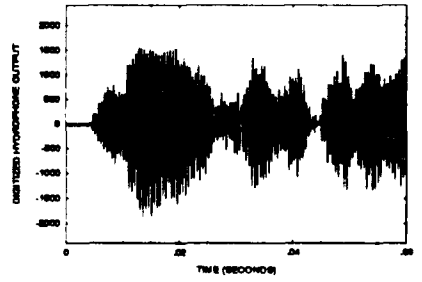
Hydrophone 5 with scattering plate



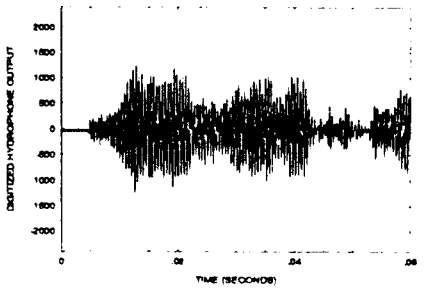
Hydrophone 5 without scattering plate



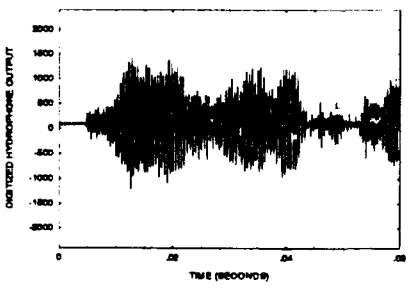
Hydrophone 6 with scattering plate



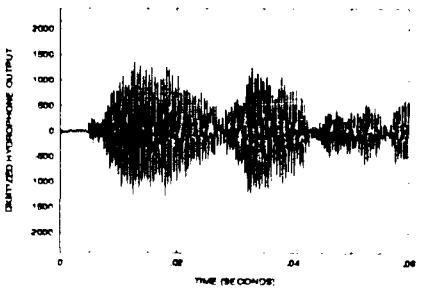
Hydrophone 6 without scattering plate



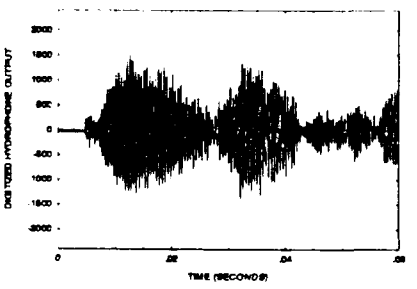
Hydrophone 7 with scattering plate



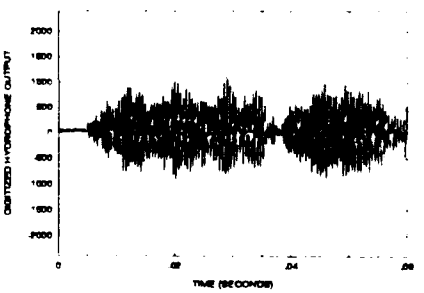
Hydrophone 7 without scattering plate



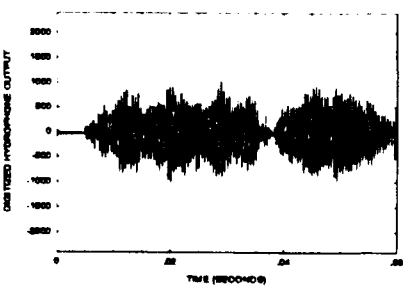
Hydrophone 8 with scattering plate



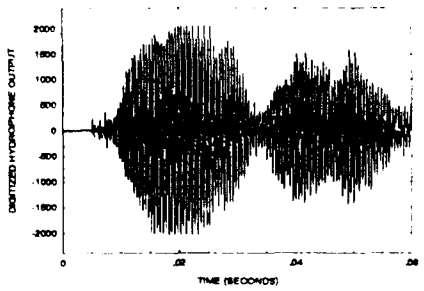
Hydrophone 8 without scattering plate



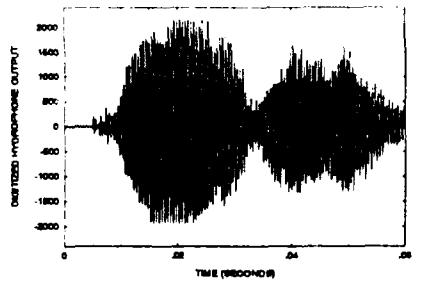
Hydrophone 9 with scattering plate



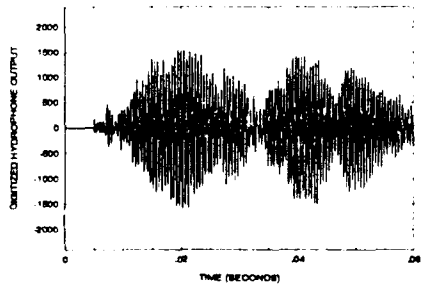
Hydrophone 9 without scattering plate



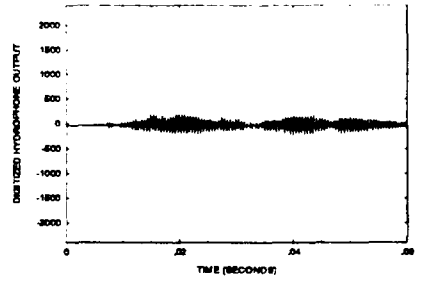
Hydrophone 10 with scattering plate



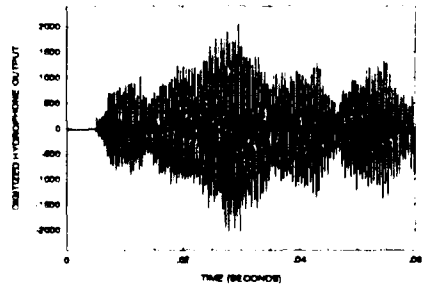
Hydrophone 10 without scattering plate



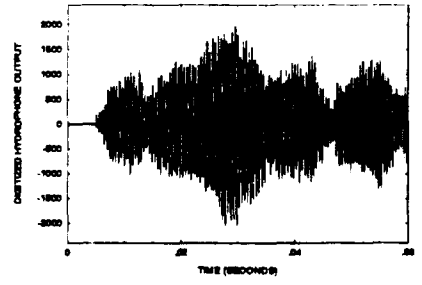
Hydrophone 11 with scattering plate



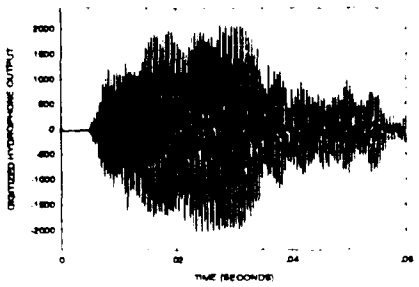
Hydrophone 11 without scattering plate



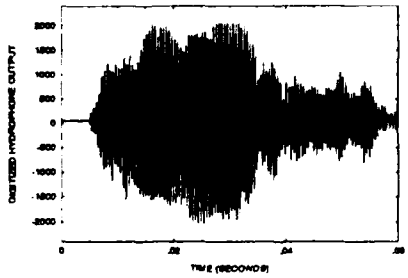
Hydrophone 12 with scattering plate



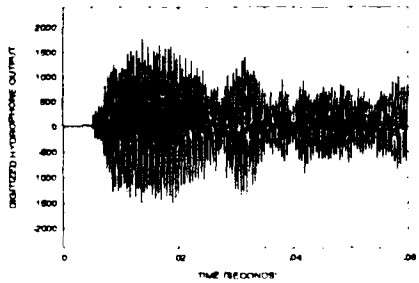
Hydrophone 12 without scattering plate



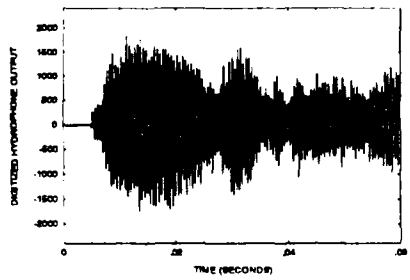
Hydrophone 13 with scattering plate



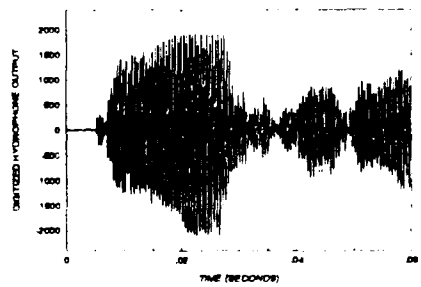
Hydrophone 13 without scattering plate



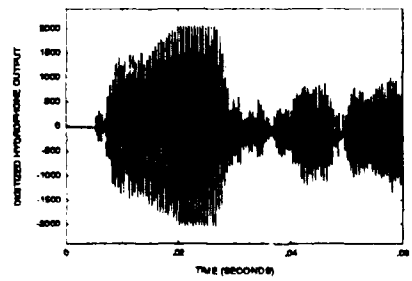
Hydrophone 14 with scattering plate



Hydrophone 14 without scattering plate

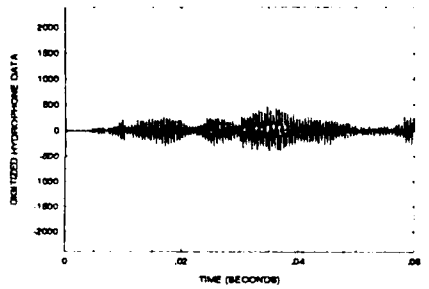


Hydrophone 15 with scattering plate

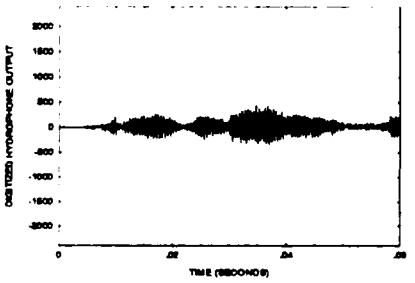


Hydrophone 15 without scattering plate

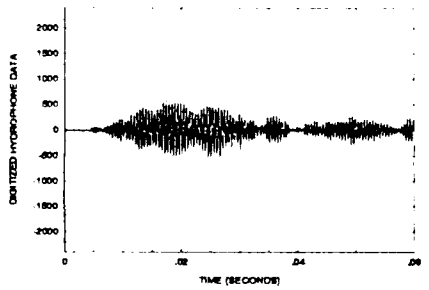
3.3 Appendix C3 Chebyshev Weighted Hydrophone Data (-30 dB)



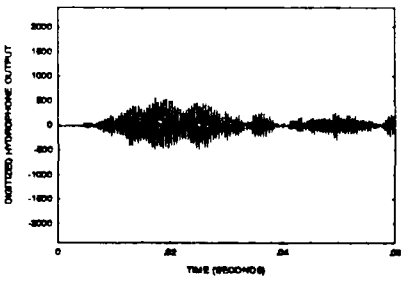
Hydrophone 1 with scattering plate



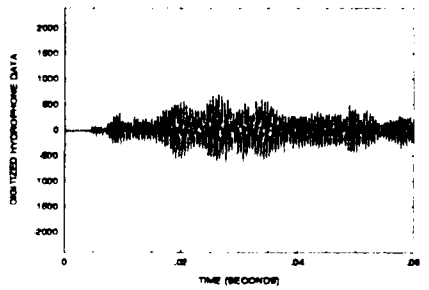
Hydrophone 1 without scattering plate



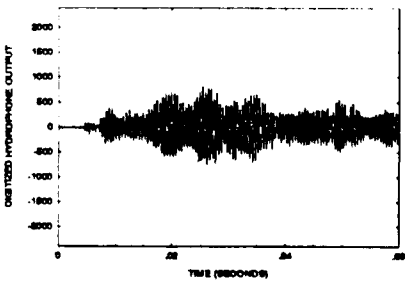
Hydrophone 2 with scattering plate



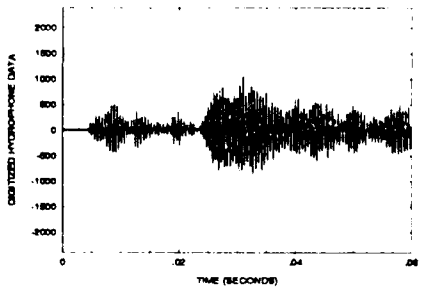
Hydrophone 2 without scattering plate



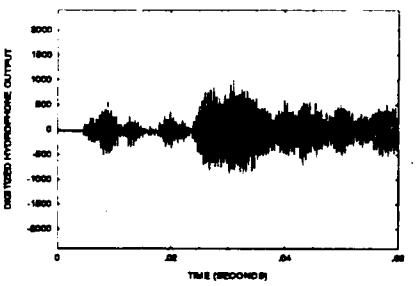
Hydrophone 3 with scattering plate



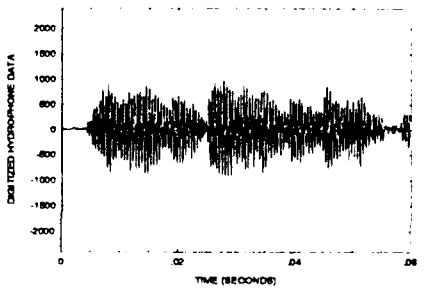
Hydrophone 3 without scattering plate



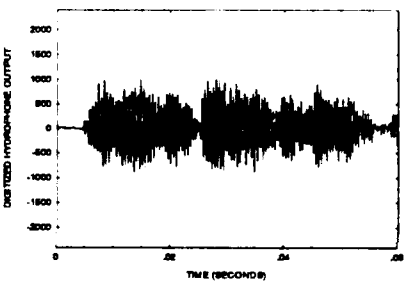
Hydrophone 4 with scattering plate



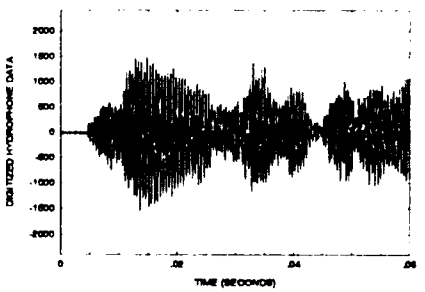
Hydrophone 4 without scattering plate



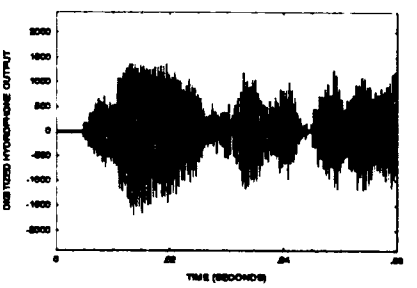
Hydrophone 5 with scattering plate



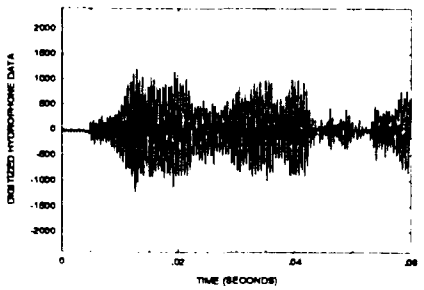
Hydrophone 5 without scattering plate



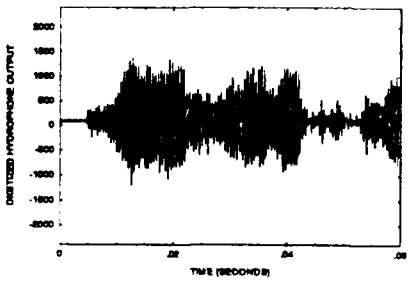
Hydrophone 6 with scattering plate



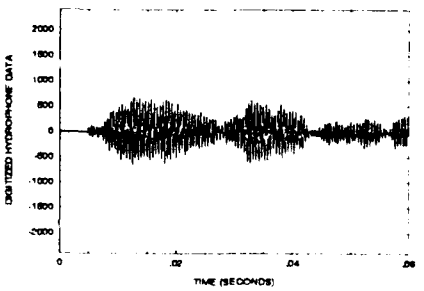
Hydrophone 6 without scattering plate



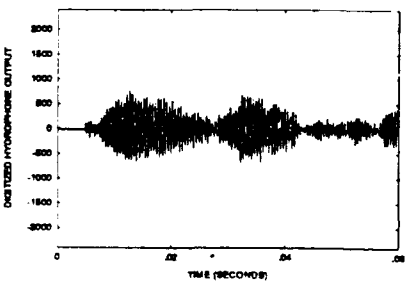
Hydrophone 7 with scattering plate



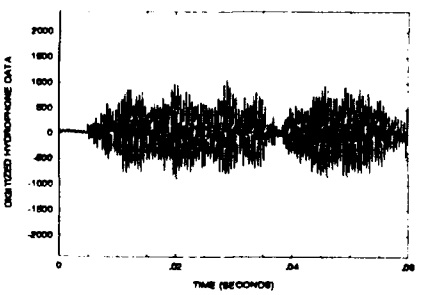
Hydrophone 7 without scattering plate



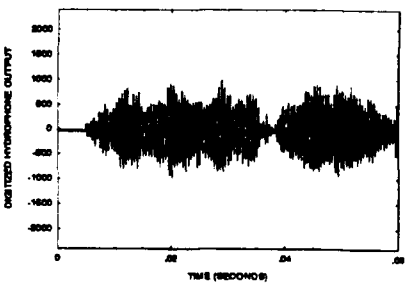
Hydrophone 8 with scattering plate



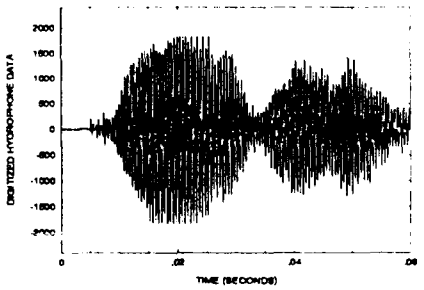
Hydrophone 8 without scattering plate



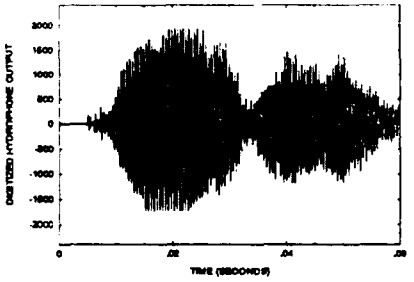
Hydrophone 9 with scattering plate



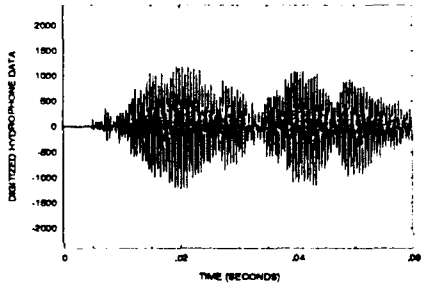
Hydrophone 9 without scattering plate



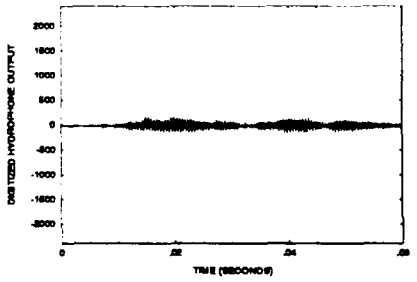
Hydrophone 10 with scattering plate



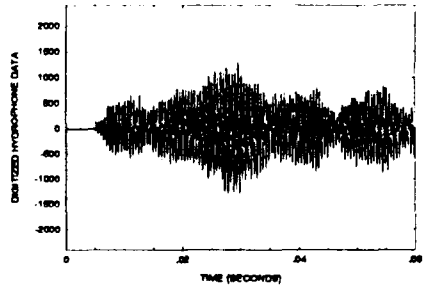
Hydrophone 10 without scattering plate



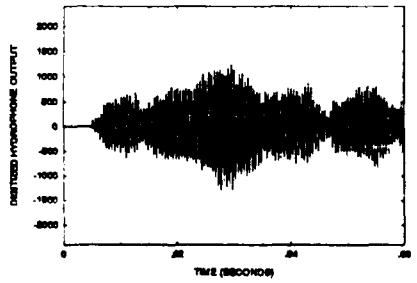
Hydrophone 11 with scattering plate



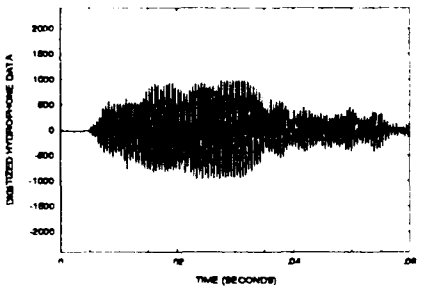
Hydrophone 11 without scattering plate



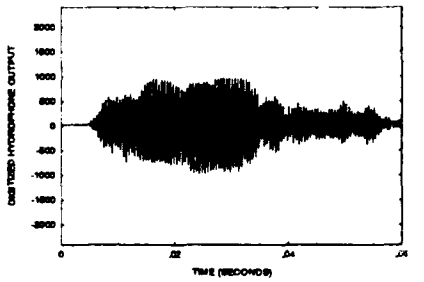
Hydrophone 12 with scattering plate



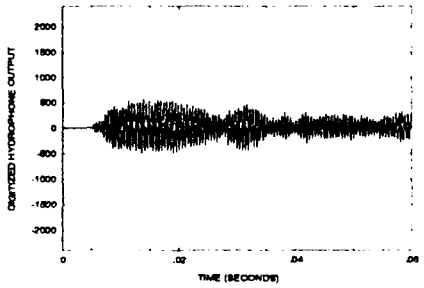
Hydrophone 12 without scattering plate



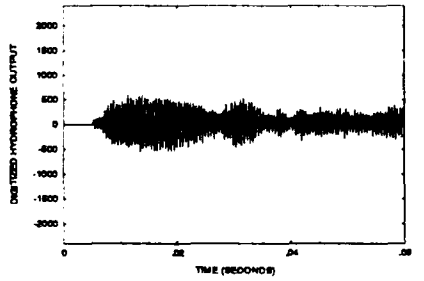
Hydrophone 13 with scattering plate



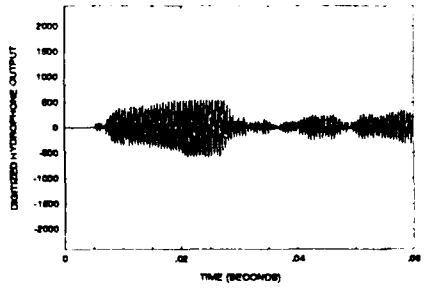
Hydrophone 13 without scattering plate



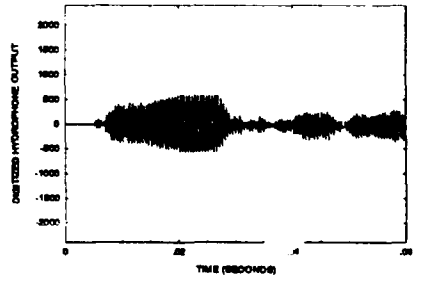
Hydrophone 14 with scattering plate



Hydrophone 14 without scattering plate



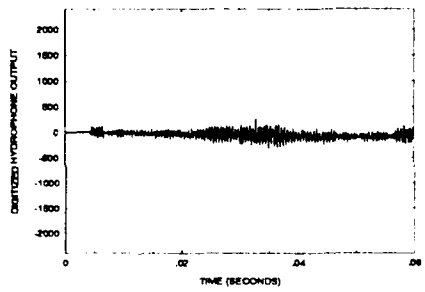
Hydrophone 15 with scattering plate



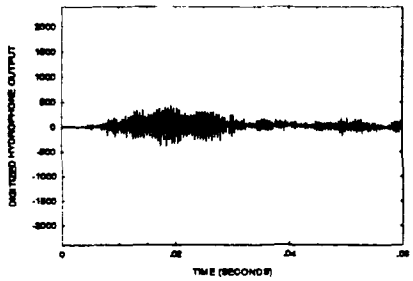
Hydrophone 15 without scattering plate

Appendix D Detecting the Scattered Signal

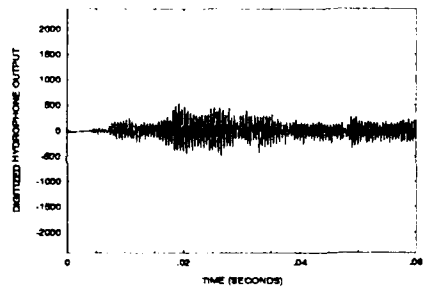
4.1 Appendix D1 Subtracting the Reverberant Field



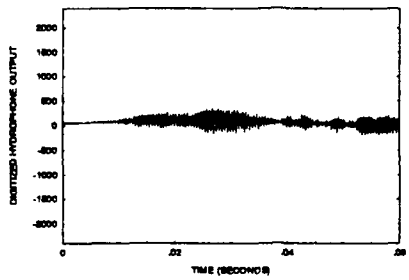
Hydrophone 1



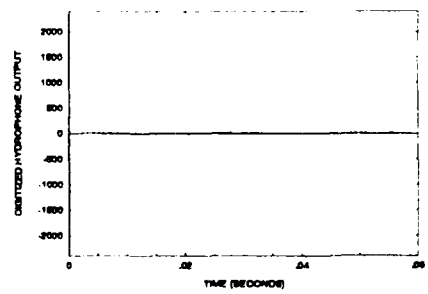
Hydrophone 2



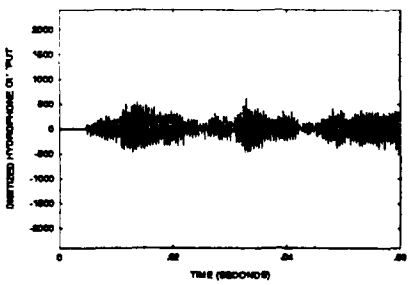
Hydrophone 3



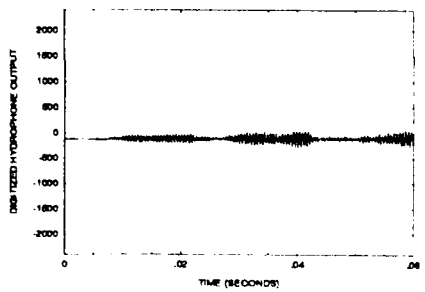
Hydrophone 4



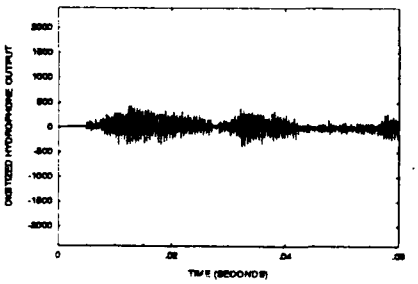
Hydrophone 5



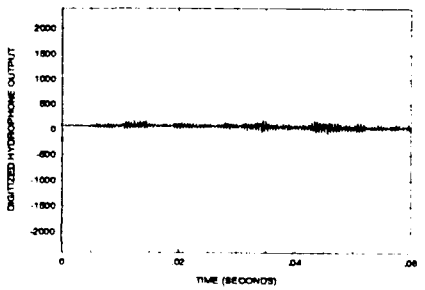
Hydrophone 6



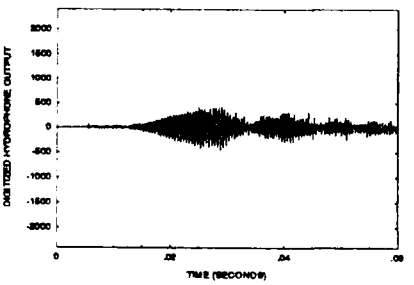
Hydrophone 7



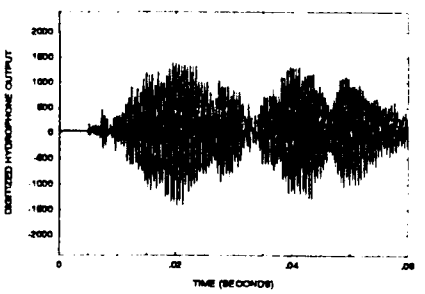
Hydrophone 8



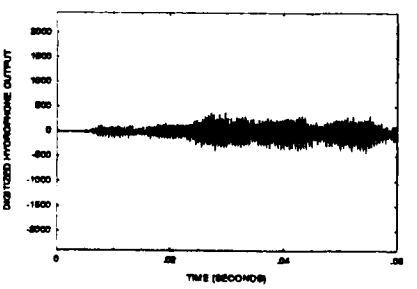
Hydrophone 9



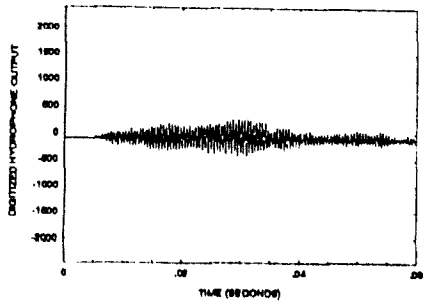
Hydrophone 10



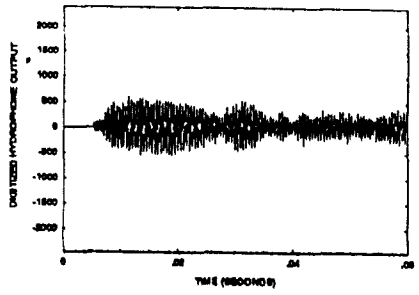
Hydrophone 11



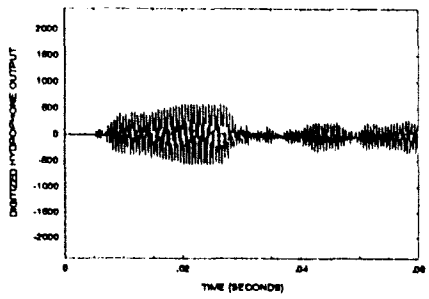
Hydrophone 12



Hydrophone 13

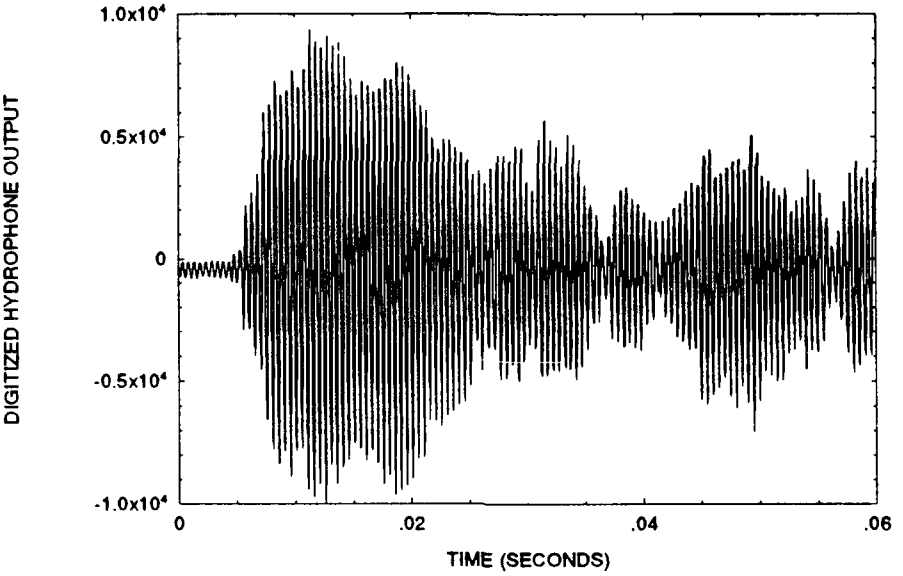


Hydrophone 14

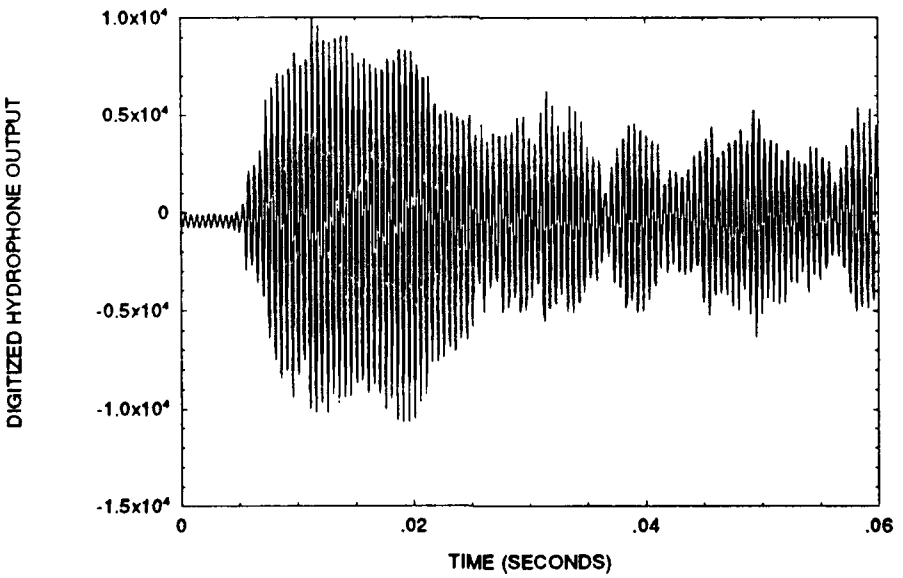


Hydrophone 15

4.2 Appendix D2 Untapered Array Beamforming

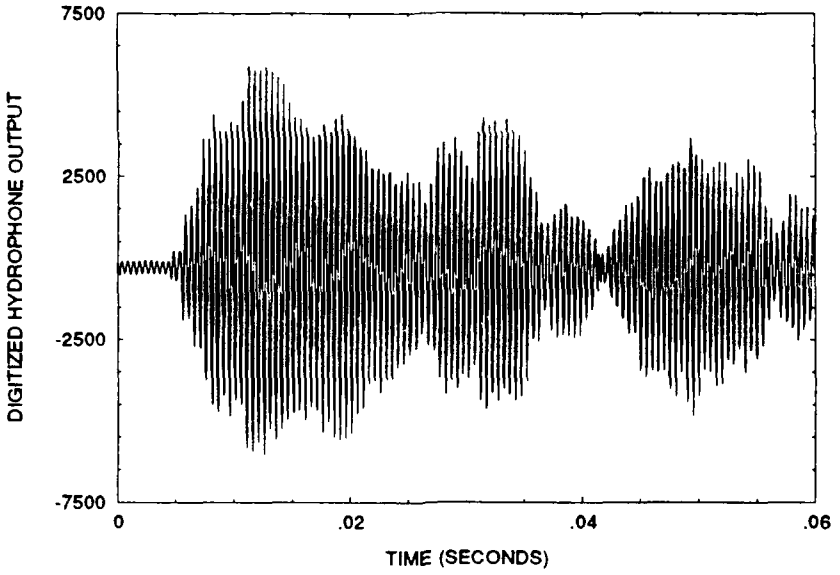


Array response with plate in the water.

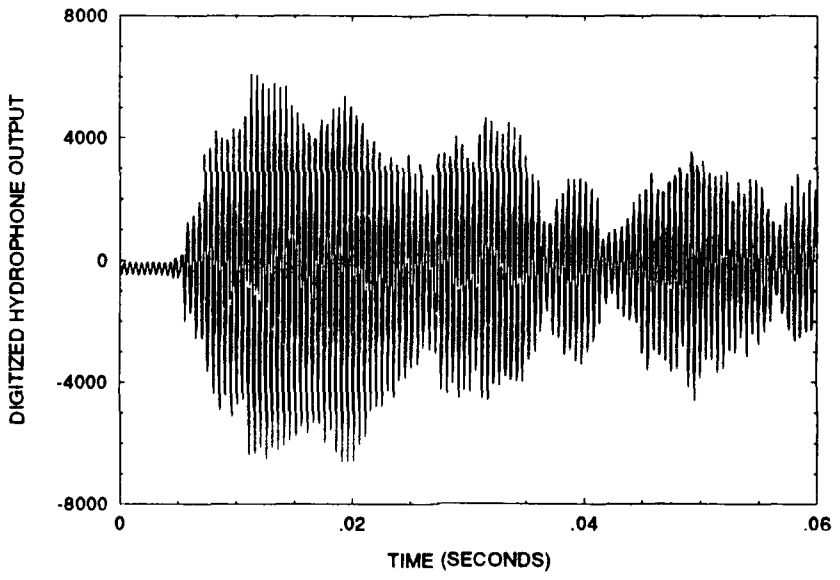


Array response with plate removed.

4.3 Appendix D3 Chebyshev Array Beamforming

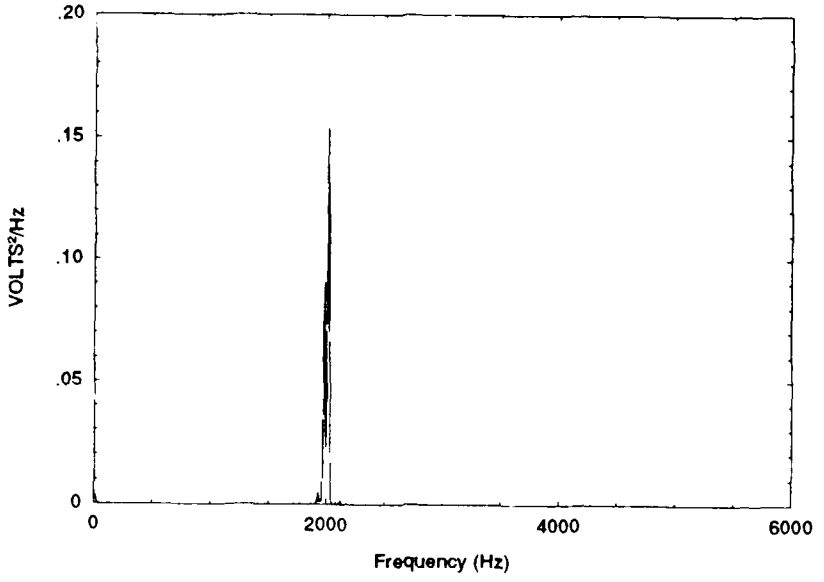


Array response with plate in the water.

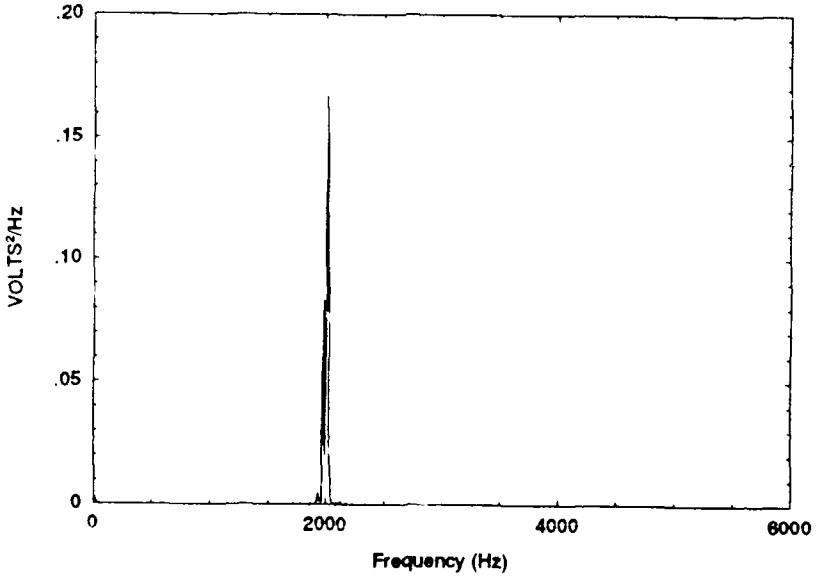


Array response with plate removed.

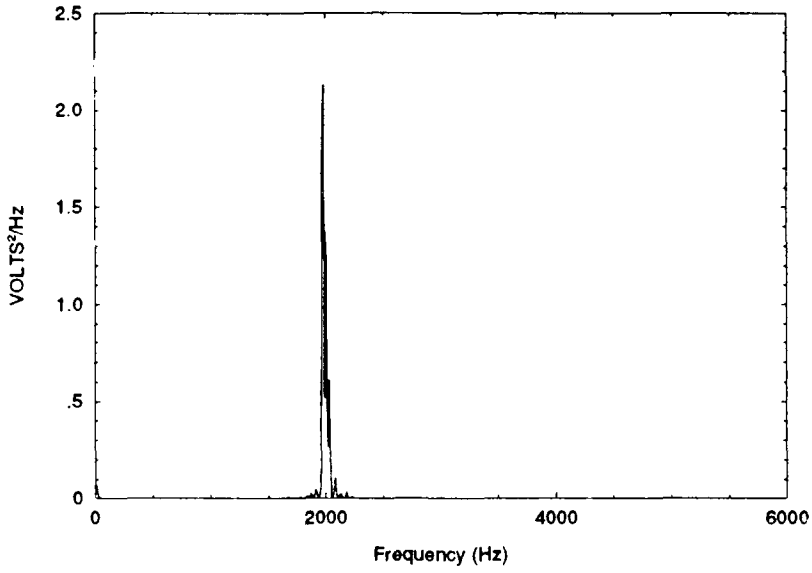
4.4 Appendix D4 Power Spectral Density Calculation



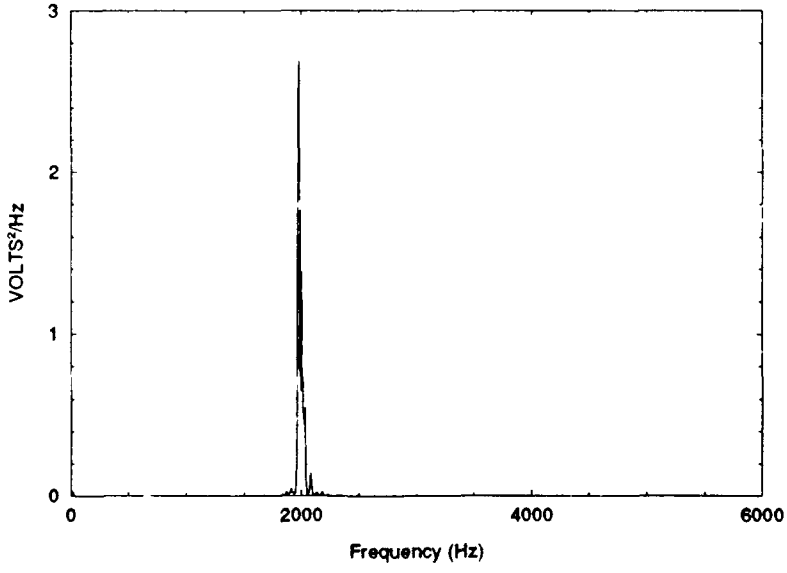
PSD omnidirectional hydrophone (#10) with plate in water.



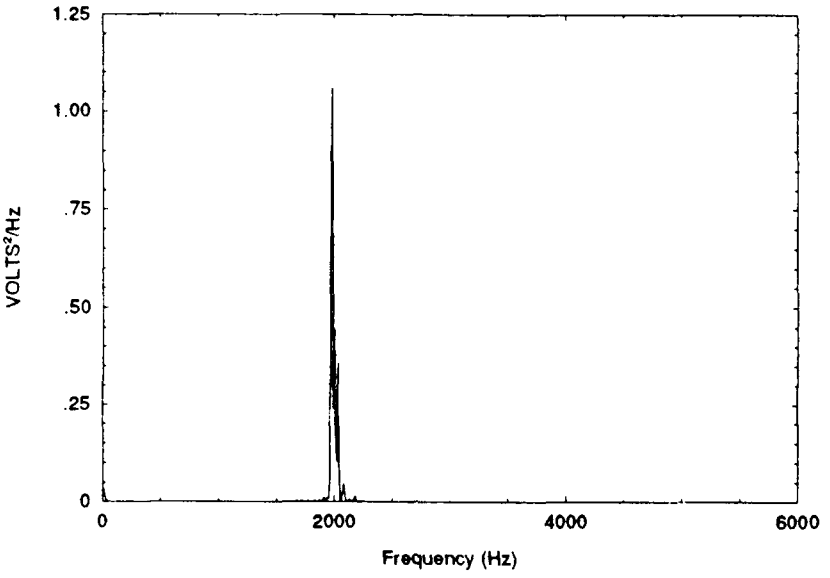
PSD omnidirectional hydrophone (#10) with plate removed.



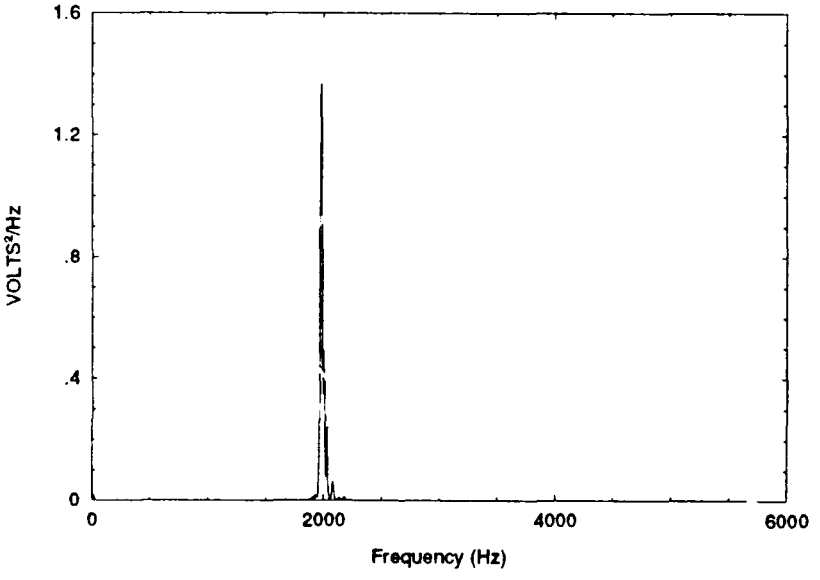
PSD RECT window with plate in water.



PSD RECT window with plate removed.



PSD Chebyshev Array with plate in water.



PSD Chebyshev Array with plate removed.